RationAnomaly: Log Anomaly Detection with Rationality via Chain-of-Thought and Reinforcement Learning

Song Xu[1,2], Yilun Liu[3]✉, Minggui He[3], Mingchen Dai[1,2], Ziang Chen[3],
Chunguang Zhao[3], Jingzhou Du[3], Shimin Tao[3], Weibin Meng[3], Mengyao Piao[3],
Shenglin Zhang[4], Yongqian Sun[4], Boxing Chen[5], Daimeng Wei[3]
[1]School of Software Engineering, University of Science and Technology of China, Hefei, China
[2]Suzhou Institute for Advanced Research, University of Science and Technology of China, Suzhou, China
[3]Huawei, Beijing, China [4]Nankai University, Tianjin, China [5]Huawei Canada, Montreal, Canada

***Abstract*—Logs serve as the foundational audit trails for monitoring the integrity and security of modern software systems. Automated log anomaly detection is crucial for ensuring the reliability of modern software infrastructures. However, existing approaches encounter significant limitations. Traditional deep learning models frequently lack interpretability and struggle with generalization, whereas Large Language Models are prone to hallucinations, introducing new security risks through factual fabrication. To address these challenges, we propose RationAnomaly, a novel framework designed to enhance log anomaly detection by synergizing Chain-of-Thought fine-tuning with reinforcement learning. Our approach initially instills expert-level reasoning patterns using supervised fine-tuning guided by Chain-of-Thought data, grounded in a high-quality dataset corrected through a rigorous expert-driven review process. Subsequently, we introduce a reinforcement learning phase employing a multi-faceted reward function to optimize the model. This function simultaneously evaluates format adherence, answer accuracy, and logical consistency, thereby effectively mitigating hallucinations. Experimental results demonstrate that RationAnomaly outperforms state-of-the-art baselines by achieving superior $\mathbf{F_1}$-scores on key benchmarks while providing transparent and step-by-step analytical outputs.**

***Index Terms*—Anomaly Detection, Fine-Tuning, Reinforcement Learning, Log Analysis, Large Language Model**

I. Introduction

Logs represent fundamental audit trails generated by software systems to reflect their operational status. Automated log analysis via machine learning has become essential for detecting anomalies and maintaining system reliability in large-scale distributed environments [1]–[3]. However, developing an accurate and trustworthy log anomaly detection system remains a challenging task.

Existing approaches predominantly fall into two categories, yet each faces significant limitations. The first category includes traditional deep learning models based on LSTMs or Transformers [4]–[6]. While these methods can capture sequential patterns, they operate as black boxes. They provide a binary classification of operational status but fail to offer semantic explanations regarding why an anomaly occurred. This lack of interpretability severely hinders operators from diagnosing root causes efficiently.

The second category leverages Large Language Models (LLMs) to enhance interpretability. Recent advancements utilize prompt engineering or Retrieval-Augmented Generation (RAG) to process log data [7]–[9]. Although RAG-based methods mitigate hallucinations by retrieving external knowledge, they introduce substantial computational overhead and retrieval latency, making them less suitable for high-throughput, real-time log streams. Furthermore, generic LLMs often lack domain-specific intuition. They are prone to hallucinations, frequently generating plausible but factually incorrect explanations. While Supervised Fine-Tuning (SFT) can adapt LLMs to the log domain [10], standard SFT maximizes the probability of the next token without explicitly validating the logical correctness of the reasoning path. Consequently, models may produce the correct label based on erroneous reasoning, a phenomenon known as the illusion of validity.

To overcome these limitations, we introduce a novel framework that internalizes expert reasoning capabilities directly into the model, thereby eliminating the need for heavy external retrieval while ensuring logical reliability. Our approach integrates Chain-of-Thought (CoT) fine-tuning with Reinforcement Learning (RL) to align the model with expert diagnostic processes.

Our contributions are summarized as follows:

- We identified that widely used public benchmarks, specifically BGL and Spirit, contain systematic labeling errors. We conducted a rigorous correction process involving industry experts to rectify these datasets. This establishes a high-fidelity benchmark that is essential for training trustworthy models.
- We propose a two-stage training paradigm to align LLM capabilities with rigorous anomaly detection requirements. A core innovation is the design of a specialized Reinforcement Learning reward function tailored for log analysis. Unlike generic text generation metrics, our reward mechanism explicitly optimizes for diagnostic accuracy, reasoning faithfulness, and factual grounding.

✉Corresponding author (liuyilun3@huawei.com).

This domain-specific feedback loop strictly penalizes hallucinations and forces the model to verify its reasoning against log semantics, ensuring high reliability in operational scenarios.

- RationAnomaly establishes comprehensive state-of-the-art performance on the BGL and Spirit benchmarks. Unlike traditional black-box methods, our framework delivers transparent and step-by-step diagnostic analysis, significantly enhancing the trustworthiness of automated operations.[1]

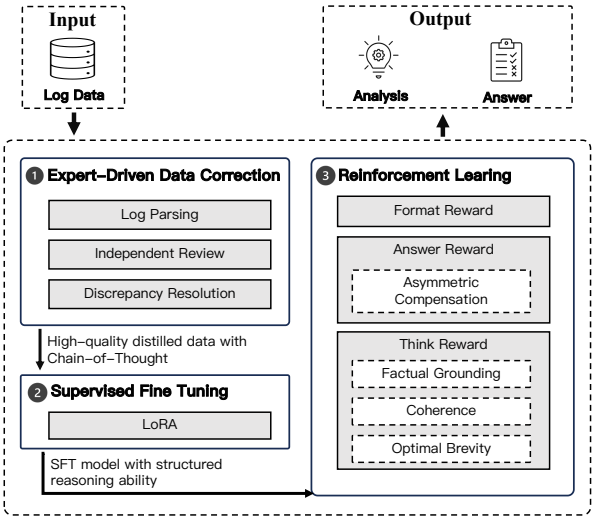

Fig. 1. Overview of RationAnomaly

II. Related Work

A. Deep Learning-based Log Anomaly Detection

Deep learning has been extensively applied to automated log analysis due to its capability to model complex patterns in sequential data [11]–[15]. Early approaches, such as DeepLog [4], utilized Long Short-Term Memory (LSTM) networks to model log sequences as natural language, identifying anomalies where predictions deviated from observed patterns. Subsequent research focused on enhancing semantic representation. LogAnomaly [5] introduced template vectors and synonym-aware word embeddings to better capture log semantics. LogRobust [6] employed attention mechanisms to handle unstable log data with varying noise levels, while CNN-based methods like SwissLog [16] utilized convolutional networks to capture local dependencies. Despite their high detection accuracy in stable environments, these discriminant models generally operate as black boxes. They output binary predictions without providing interpretable insights into the root cause, which limits their utility in practical troubleshooting scenarios where explainability is paramount.

[1]Source code and the expert-corrected datasets can be found at the project repository: https://github.com/Gravityless/RationAnomaly.

B. LLM-based Log Anomaly Detection

The emergence of Large Language Models has introduced new paradigms for log analysis, primarily categorized into prompt-based and fine-tuning-based approaches [17]–[20].

1) Prompt-based and RAG Methods: Prompt-based methods leverage the zero-shot or few-shot capabilities of LLMs. LogPrompt [7] employs in-context learning to guide LLMs in identifying anomalies. To address the hallucination issues inherent in generic LLMs, recent studies have integrated Retrieval-Augmented Generation. Approaches such as LogRAG [8], RAGLog [9], and EagerLog [21] retrieve relevant knowledge or historical log patterns from an external database to ground the model's generation. While these methods improve factual accuracy, they introduce dependency on external retrieval modules. This dependency results in increased computational latency and storage overhead during inference, posing challenges for real-time anomaly detection in high-frequency log streams.

2) Fine-tuning Methods: Fine-tuning adapts LLMs specifically to the log domain. LogGPT [10] applies supervised fine-tuning to train models on log sequences, enabling them to predict subsequent log entries. LogLM [22] and SuperLog [23] further explore instruction tuning to improve the model's ability to follow diagnostic commands. However, these methods typically employ standard Supervised Fine-Tuning loss, which optimizes token-level likelihood rather than reasoning quality. Consequently, they do not explicitly enforce the logical consistency of the diagnostic process.

C. Reasoning Alignment in Logs

Our work distinguishes itself by addressing the gap between generation probability and reasoning correctness. Instead of relying on external retrieval or standard SFT, we introduce a Reinforcement Learning Alignment phase. This phase explicitly optimizes the model to follow a verified Chain-of-Thought, ensuring that the detected anomalies are supported by sound, expert-aligned reasoning internal to the model.

III. Methodology

A. Overview of RationAnomaly

We propose RationAnomaly, a framework designed to empower Large Language Models with the capability to perform expert-level log anomaly detection without relying on computationally expensive external retrieval systems. As illustrated in Fig. 1, the training pipeline comprises three sequential phases. First, we construct a high-quality instruction dataset incorporating Chain-of-Thought reasoning paths. Second, we utilize this dataset for Supervised Fine-Tuning to instill rudimentary domain knowledge and reasoning patterns into the base model. Third, to address the illusion of validity where models generate correct labels via erroneous logic, we implement a Reinforcement Learning alignment phase. We employ Group Relative Policy Optimization (GRPO) to refine the model, utilizing a composite reward function that evaluates format compliance, answer accuracy, and logical consistency.

TABLE I
DETAILED BREAKDOWN OF ANNOTATION ERRORS IDENTIFIED AND CORRECTED IN THE DATASET.

Correction	Error Category	Count	Percentage	Representative Examples
Normal → Abnormal	System Error	78	34.7%	"PANIC: segmentation violation..." "hit ASSERT condition..."
	Network Issue	47	20.9%	"Connection refused..." "Bad UMNT RPC: RPC: Timed out"
	Hardware Failure	40	17.8%	"parity error in read queue..." "DDR failing info register..."
	Software Exception	56	24.9%	"ciod: Error loading..." "divide-by-zero..."
Abnormal → Normal	-	4	1.8%	"exited normally with exit code..." "Mounting NFS filesystems..."
Total	**-**	**225**	**100.0%**	**-**

B. Phase I: Expert-Driven Data Correction

The reliability of any data-driven model is contingent upon the quality of its training and evaluation data. Public benchmarks like BGL and Spirit, while widely used [24], contain systematic labeling errors that can compromise model evaluation. To establish a trustworthy foundation, we conducted a comprehensive data correction process.

We assembled a team of five industry experts to systematically review all 3,046 unique log templates extracted from the BGL and Spirit datasets. The process began with an independent review by each expert. Discrepancies were then resolved through consensus-driven panel discussions, with a senior expert providing final validation for disputed cases. As shown in Table I, this rigorous process identified and corrected 225 systematically mislabeled log templates (7.4% of total). Our analysis revealed a significant bias towards false negatives (98.2% of errors), where critical system failures (e.g., "segmentation violation", "Connection refused") were incorrectly marked as normal. The result is a high-fidelity benchmark characterized by inter-annotator agreement at κ=0.94, forming a reliable basis for our subsequent training and evaluation.

This correction process reveals critical insights into the limitations of current benchmarks. As detailed in Table I, "System Error" accounts for the highest proportion (34.7%) of mislabeled instances. A closer examination indicates that these logs often contain keywords like "PANIC" or "violation" involving low-level kernel routines. Previous automated labeling scripts or non-expert annotators likely categorized them based on simple keyword matches or generic severity levels rather than understanding the fatal implications of these kernel states. For instance, a "segmentation violation" in a critical daemon is a system-halting event, yet was frequently labeled as "Normal" in the original BGL dataset.

The high prevalence of False Negatives (Normal → Abnormal) in the original dataset has profound implications for previous research. It suggests that prior SOTA methods achieving near-perfect scores on unmodified BGL/Spirit might have been overfitting to noise—essentially learning to ignore critical system failures because the ground truth demanded it. By correcting these 225 systematic errors, we do more than just improve dataset hygiene; we recalibrate the benchmark to penalize models that overlook silent failures. This contribution ensures that future progress on these benchmarks reflects genuine improvements in anomaly detection capability rather than the ability to mimic historical labeling biases.

C. Phase II: CoT-Guided Supervised Fine-Tuning

The primary objective of this phase is to adapt a generic LLM to the specific linguistic and structural characteristics of system logs. Unlike standard instruction tuning that pairs an input log x directly with a label y, we introduce an intermediate reasoning step r. The training data consists of triplets (x, r, y), where r explicitly articulates the step-by-step analysis performed by a human expert.

We curate a novel dataset based on the corrected BGL and Spirit benchmarks. Specifically, we leverage GPT-4o [25] to generate initial Chain-of-Thought reasoning paths for each log entry, covering template analysis, severity level identification, and variable anomaly checking. All generated reasoning paths were then verified and corrected through expert review to ensure factual accuracy and logical consistency. The optimization objective for SFT is to minimize the standard negative log-likelihood loss:

$$\mathcal{L}_{SFT}(\theta) = -\sum_{t=1}^{T} \log P_\theta(u_t|x, u_{<t}) \tag{1}$$

where θ represents the model parameters, x is the input log sequence, and $u = [r; y]$ denotes the concatenation of the reasoning path and the final prediction. This phase serves as a "warm-up" strategy, ensuring the model's policy π_θ initializes in a viable region of the solution space before reinforcement learning begins.

D. Phase III: Reinforcement Learning Alignment

While SFT teaches the model *what* to say, it does not guarantee the logical validity of the generation. To strictly align the model's reasoning with factual evidence, we employ Reinforcement Learning.

1) Group Relative Policy Optimization (GRPO): We adopt Group Relative Policy Optimization [26], a simplified variant of Proximal Policy Optimization (PPO) that eliminates the need for a separate value network (Critic). This significantly reduces memory overhead and training instability. For each input prompt q, the model generates a group of G outputs $\{o_1, o_2, ..., o_G\}$. We calculate the reward R_i for each output using our reward model. To stabilize training, we normalize the raw rewards within the group. The advantage A_i for the i-th output is computed as:

$$A_i = \frac{R_i - \text{mean}(\{R_1, ..., R_G\})}{\text{std}(\{R_1, ..., R_G\}) + \epsilon} \tag{2}$$

The GRPO objective function is then defined as:

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}_{q\sim D, \{o_i\}_{i=1}^{G}\sim\pi_{\theta_{old}}} \left[\frac{1}{G}\sum_{i=1}^{G} \min\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)} A_i, \text{clip}(\dots) A_i \right) - \beta \mathbb{D}_{KL} \right] \tag{3}$$

where β controls the KL-divergence penalty to prevent the model from deviating excessively from the reference SFT model.

2) Multi-Faceted Reward Design: The core of our alignment strategy lies in the design of the reward function R. We decompose the reward into three distinct components to conduct multi-dimensional supervision:

$$R = \lambda_1 R_{\text{format}} + \lambda_2 R_{\text{think}} + \lambda_3 R_{\text{answer}} \tag{4}$$

where λ constitutes the hyperparameters balancing the contribution of each component.

Format Reward (R_{format}): To ensure parsability and structural integrity, we define strict delimiters. Let S be the generated sequence. The reward is modeled as a binary indicator function:

$$R_{\text{format}} = \mathbb{I}(S \text{ contains } \texttt{<think>}, \texttt{</think>}, \texttt{<answer>}) \tag{5}$$

where $\mathbb{I}(\cdot)$ equals 1 if the condition is met and -1 otherwise. Crucially, if $R_{\text{format}} = -1$, we force $R_{\text{think}} = R_{\text{answer}} = 0$ to strictly penalize malformed outputs and prevent the model from gaming the system by generating correct answers in invalid formats.

Answer Reward (R_{answer}): Log data is inherently imbalanced, and in production environments, False Negatives (missing an anomaly) are significantly more costly than False Positives. We introduce an asymmetric reward mechanism. Let $y \in \{0, 1\}$ be the ground truth label (1 for anomaly) and $\hat{y}$ be the prediction:

$$R_{\text{answer}} = \begin{cases} +\alpha & \text{if } \hat{y} = y \text{ and } y = 1 \quad \text{(True Positive)} \\ +1 & \text{if } \hat{y} = y \text{ and } y = 0 \quad \text{(True Negative)} \\ -\alpha & \text{if } \hat{y} \neq y \text{ and } y = 1 \quad \text{(False Negative)} \\ -1 & \text{if } \hat{y} \neq y \text{ and } y = 0 \quad \text{(False Positive)} \end{cases} \tag{6}$$

Here, $\alpha > 1$ is a weighting factor (e.g., $\alpha = 2.5$) that incentivizes the model to be more sensitive to anomalies.

Thinking Reward (R_{think}): To combat hallucination and ensure high-quality reasoning, we optimize the intermediate thought sequence o_{think} along three dimensions:

$$R_{\text{think}} = \gamma_1 S_{\text{ground}} + \gamma_2 S_{\text{cohere}} + \gamma_3 S_{\text{brev}} \tag{7}$$

Factual Grounding (S_{ground}): To penalize fabricated evidence, we measure the semantic overlap between the model's generated reasoning o_{think} and the teacher's reasoning o_{ref} from our distilled CoT dataset. This serves as a robust and scalable proxy for factual grounding against a known-good reasoning path, thus effectively discouraging hallucination. We employ a harmonic mean of BLEU and ROUGE-L scores:

$$S_{\text{ground}} = \frac{2 \cdot \text{BLEU}(o_{think}, o_{ref}) \cdot \text{ROUGE}(o_{think}, o_{ref})}{\text{BLEU}(o_{think}, o_{ref}) + \text{ROUGE}(o_{think}, o_{ref}) + \epsilon} \tag{8}$$

Coherence (S_{cohere}): We utilize a lightweight perplexity model (PPL) to evaluate fluency. The score decays with higher perplexity to punish repetitive or nonsensical generation:

$$S_{\text{cohere}} = -\frac{\text{PPL}(o_{think})}{\tau} \tag{9}$$

where τ is a scaling hyperparameter.

Optimal Brevity (S_{brev}): Instead of enforcing a global length constraint, we align the complexity of the generated reasoning with the specific difficulty of the log entry. We compare the length of the generated thought sequence $|o_{think}|$ against the length of the distilled teacher reasoning $|o_{ref}|$ for that specific instance. The reward is formulated as a quadratic penalty:

$$S_{\text{brev}} = -\frac{(|o_{think}| - |o_{ref}|)^2}{\sigma_{len}^2} \tag{10}$$

where $|\cdot|$ denotes the token count and σ_{len} is a scaling factor controlling the tolerance for length deviation. This mechanism encourages the model to mimic the expert's conciseness: providing detailed analysis for complex logs while remaining brief for obvious cases.

E. Inference Efficiency Analysis

A critical requirement for log anomaly detection in production environments is low and predictable latency. In the Introduction, we identified that RAG-based methods (e.g., RAGLog [9]) suffer from computational overhead due to external retrieval. Here, we formally analyze the theoretical efficiency of RationAnomaly compared to these approaches to demonstrate our architectural advantage.

TABLE II
OVERALL RESULT

Method	BGL(Session-level)			BGL(Template-level)			Spirit(Session-level)			Spirit(Template-level)		
	F_1	Pre	Rec	F_1	Pre	Rec	F_1	Pre	Rec	F_1	Pre	Rec
DeepLog	0.869	0.811	0.935	-	-	-	0.905	0.891	0.919	-	-	-
LogAnomaly	0.857	0.770	**0.965**	-	-	-	0.925	0.880	0.975	-	-	-
LogRobust	0.811	0.704	0.956	-	-	-	0.921	0.856	**0.996**	-	-	-
LogPrompt	0.834	0.874	0.811	0.827	0.724	**0.964**	0.917	0.859	0.983	0.791	0.734	**0.858**
LogLM	0.799	0.842	0.761	0.847	0.833	0.861	0.915	0.871	0.964	0.801	0.856	0.754
GPT-4o	0.792	0.821	0.765	0.769	0.835	0.714	0.877	0.863	0.891	0.687	0.769	0.621
Llama 2 7B	0.707	0.741	0.753	0.686	0.858	0.633	0.800	0.881	0.861	0.655	0.839	0.598
RationAnomaly	**0.909**	**0.900**	0.919	**0.887**	**0.898**	0.881	**0.958**	**0.959**	0.959	**0.862**	**0.899**	0.847

The inference latency of a typical RAG-based system, T_{RAG}, can be decomposed into three stages: embedding, retrieval, and generation.

$$T_{RAG} = T_{emb}(x) + T_{search}(\mathcal{D}) + T_{gen}(x, C) \tag{11}$$

where T_{emb} is the time to encode the query log x, T_{search} represents the vector similarity search over an external database $\mathcal{D}$, and T_{gen} is the LLM generation time conditioned on both x and the retrieved context C. The bottleneck lies in $T_{search}(\mathcal{D})$. As the size of historical logs $|\mathcal{D}|$ grows, retrieval latency typically increases (scaling logarithmically $O(\log |\mathcal{D}|)$ or linearly $O(|\mathcal{D}|)$ depending on indexing). Furthermore, RAG introduces significantly higher memory bandwidth interaction and I/O latency.

In contrast, RationAnomaly eliminates the dependency on external knowledge bases by internalizing expert reasoning patterns directly into the model parameters θ through CoT-SFT and RL. Our inference latency $T_{RationAnomaly}$ is formally defined as:

$$T_{RationAnomaly} = T_{gen}(x, \hat{r}) \tag{12}$$

where $\hat{r}$ is the internally generated reasoning path. Crucially, our method removes both the embedding step T_{emb} and the retrieval step T_{search}. Hence, the time complexity of RationAnomaly is $O(1)$ with respect to the volume of historical data $|\mathcal{D}|$. This independence ensures that RationAnomaly delivers stable, deterministic latency, making it significantly more robust for high-throughput real-time log streams compared to retrieval-dependent architectures.

IV. EXPERIMENTS

A. Experimental Setup

1) Datasets: For the BGL and Spirit datasets, we performed a chronological split to simulate real-world data flow. To support deep learning baselines, we first sampled a 2000-log template-level training set (15% anomaly rate), then constructed session-level data using a 100-log fixed window [27]. Test sets were created similarly, resulting in 8000 entries and sessions per dataset. To prevent data leakage, logs used in RationAnomaly's training were excluded from these session-level test sets. To ensure a fair comparison, all baseline methods were re-trained and evaluated using our corrected datasets.

2) Baselines: We compare RationAnomaly against two categories of state-of-the-art methods: conventional deep learning models, including unsupervised approaches like DeepLog [4] and LogAnomaly [5] as well as the supervised LogRobust [6]; and LLM-based techniques, such as LogPrompt [7], LogLM [22], GPT-4o [25] and Llama 2 7B [28] baseline.

3) Implementation Details: Our backbone model is Llama 2 7B, chosen for its balance of performance and inference efficiency. We performed SFT for 3 epochs with a learning rate of $4e^{-7}$. For reinforcement learning, we utilized the GRPO algorithm with a group size $G = 8$, a learning rate of $4e^{-7}$, and trained for 2 epochs. The KL-divergence penalty was set to $\beta = 0.001$. The reward coefficients were set to $\lambda_1 = 0.2$ (format), $\lambda_2 = 0.5$ (reasoning validity), and $\lambda_3 = 0.5$ (answer correctness). For the Thinking Reward, the sub-component weights were $\gamma_1 = 0.5$ (factual grounding), $\gamma_2 = 1.0$ (coherence), and $\gamma_3 = 0.5$ (brevity), with the coherence scaling factor $\tau = 50$ and the brevity tolerance $\sigma_{len} = 10$. The coherence score was computed using DistilGPT-2 as the lightweight perplexity model. All experiments were conducted on a server equipped with 8 $\times$ NVIDIA A100 GPUs.

B. Experimental Results

We evaluate performance using Precision, Recall, and F_1-Score. Table II presents a comprehensive performance comparison. RationAnomaly establishes a new state-of-the-art, achieving superior F_1-scores across all evaluated datasets and scenarios.

Comparison with Deep Learning Baselines. As shown in Table II, RationAnomaly surpasses conventional deep learning baselines across all settings. On session-level detection, our method achieves an **F_1-score of 0.958** on Spirit, a notable improvement over the best-performing traditional baseline, LogAnomaly (0.925). Furthermore, unlike traditional methods (e.g., DeepLog, LogAnomaly) which typically rely on index-based parsing and are often limited to session-level analysis, RationAnomaly's semantic understanding enables effective template-level detection. This is evidenced by F_1-scores of 0.887 on BGL and 0.862 on Spirit, confirming its capability for fine-grained interpretation beyond simple sequence matching.

Comparison with LLM-based Approaches. We further benchmark RationAnomaly against leading LLM-based methods, LogLM, and the general-purpose model, GPT-4o. **Ratio-**

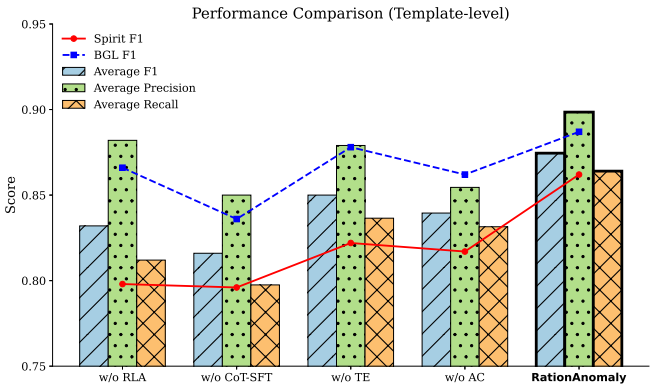

Fig. 2. Ablation Result

nAnomaly significantly outperforms GPT-4o despite having orders of magnitude fewer parameters. For instance, on the Spirit (Template-level) task, GPT-4o only achieves an F_1-score of 0.687, falling far behind our 0.862. A closer look at the metrics reveals that GPT-4o suffers from low sensitivity (e.g., 0.621 Recall on Spirit Template), suggesting that without domain-specific fine-tuning, general LLMs often lack the specialized knowledge to recognize obscure kernel errors or are overly conservative in their judgments. Similarly, while LogLM improves upon the zero-shot baseline, it consistently lags behind RationAnomaly (e.g., 0.799 vs. 0.909 on BGL Session). This empirically validates that simply adapting an LLM to logs is insufficient; the explicit **Reasoning Alignment (CoT)** and **RM-guided optimization** are the key factors that enable RationAnomaly to bridge the gap between general language capability and precise anomaly detection.

The performance delta between RationAnomaly and the zero-shot Llama 2 7B baseline highlights the necessity of our training paradigm. On BGL template-level, **the F_1-score shows a 29.3% relative improvement** over the base model. Moreover, a critical outcome of our method is the achievement of a **well-balanced precision-recall profile**. While traditional methods like LogRobust often trade Precision for Recall (e.g., 0.704 Pre vs. 0.956 Rec), RationAnomaly attains 0.959 for both metrics on the Spirit session dataset. This balance is a direct consequence of the Reinforcement Learning Alignment stage, which penalizes hallucinations (improving Precision) while encouraging the detection of subtle faults (maintaining Recall), making it highly reliable for operational deployment.

C. Ablation Study

To dissect the contribution of each component within our framework, we conducted an ablation study. As shown in Fig. 2, the ablation experiment results show the same trend on BGL and Spirit datasets.

The model without CoT-SFT solely undergoes reinforcement learning achieves the lowest F_1-score and significantly lower Precision and Recall values. This indicates that fine-tuning on a structured reasoning process provides an important and effective foundation for subsequent steps.

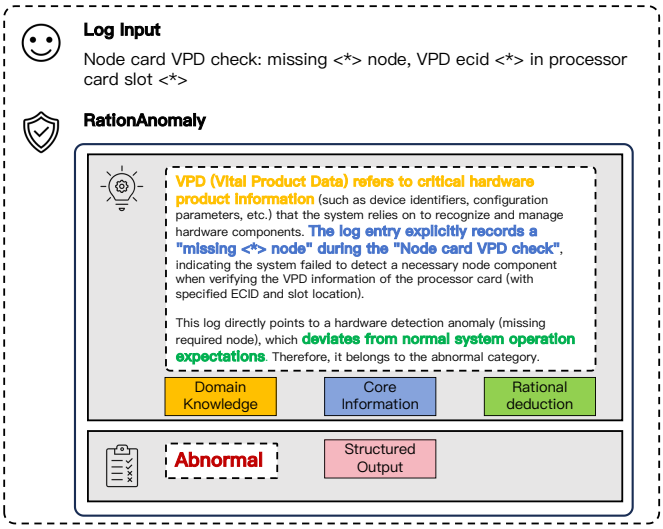

Fig. 3. Qualitative case study illustrating the interpretability of RationAnomaly. It first leverages domain knowledge to define terms, extracts core symptoms, and performs rational deduction to identify the root cause. This step-by-step visibility builds trust in automated diagnostics.

The necessity of RLA stage is made evident by the significant performance increase from the model without RLA to the final RationAnomaly model, which is reflected in **F_1-score improvement from 0.798 to 0.862** on Spirit. This result indicates that RLA is an essential step for refining the model's behavior, correcting subtle reasoning flaws, and aligning its outputs toward maximal accuracy.

The specific design of the reward function proves to be pivotal. Disabling the Asymmetric Compensation (without AC) or the Thinking Evaluation (without TE) leads to distinct performance degradation, with **F_1-scores dropping from 0.862 to 0.817 and 0.822** on Spirit. This confirms that both applying greater penalties to high-cost errors and rewarding factually-grounded reasoning are critical mechanisms for achieving the model's final, robust performance.

D. Case Study

To demonstrate our model's capabilities, we present a case study in Fig. 3. The log entry requires domain-specific knowledge that challenges conventional methods. RationAnomaly not only correctly classifies it as abnormal but also generates a transparent, expert-like rationale. It leverages its acquired **Domain Knowledge** to define VPD, extracts the **Core Information** ("missing node"), and performs a **Rational Deduction** to identify a hardware detection failure. This structured output, a direct result of our CoT-SFT and RLA pipeline, showcases a shift from simple pattern matching to providing verifiable and trustworthy diagnostic insights.

V. Conclusion and Future Work

In this paper, we introduced RationAnomaly, a novel framework that bridges the gap between Large Language Model generation and reliable log anomaly detection. By integrating Chain-of-Thought guided Supervised Fine-Tuning with

a reinforcement learning alignment phase based on Group Relative Policy Optimization, we have successfully addressed the critical issue of the illusion of validity in LLM outputs.

Our extensive experiments on expert-corrected BGL and Spirit datasets demonstrate that RationAnomaly not only achieves state-of-the-art detection accuracy, surpassing both traditional deep learning models and standard prompt-based LLMs, but also provides transparent, expert-aligned reasoning for its decisions. Crucially, our approach eliminates the dependency on external retrieval modules inherent in RAG-based methods, offering a more self-contained and architecturally streamlined solution. The ablation studies confirm that our multi-faceted reward mechanism effectively minimizes reasoning hallucinations and reduces false positive rates.

For future work, we plan to extend this framework to online learning settings, enabling the model to adapt continuously to evolving log patterns without catastrophic forgetting. Additionally, we aim to explore quantization techniques to further reduce the inference latency, making deployment on edge devices feasible for real-time monitoring in large-scale distributed systems.

REFERENCES

[1] Shuyu Lin, Ronald Clark, Robert Birke, Sandro Schönborn, Niki Trigoni, and Stephen Roberts, "Anomaly detection for time series using vae-lstm hybrid model," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2020, pp. 4322–4326.

[2] Lingzhe Zhang, Tong Jia, Mengxi Jia, Hongyi Liu, Yong Yang, Zhonghai Wu, and Ying Li, "Towards close-to-zero runtime collection overhead: Raft-based anomaly diagnosis on system faults for distributed storage system," *IEEE Transactions on Services Computing*, vol. 18, no. 2, pp. 1054–1067, 2025.

[3] Lingzhe Zhang, Tong Jia, Mengxi Jia, Yifan Wu, Aiwei Liu, Yong Yang, Zhonghai Wu, Xuming Hu, Philip Yu, and Ying Li, "A survey of aiops in the era of large language models," *ACM Comput. Surv.*, vol. 58, no. 2, Sept. 2025.

[4] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2017, pp. 1285–1298, ACM.

[5] Weibin Meng, Ying Liu, Yichen Zhu, Shenglin Zhang, Dan Pei, Yuqing Liu, Yihao Chen, Ruizhi Zhang, Shimin Tao, Pei Sun, and Rong Zhou, "Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. 2019, pp. 4739–4745, ijcai.org.

[6] 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, "Robust log-based anomaly detection on unstable log data," in *ESEC/SIGSOFT FSE 2019*. 2019, pp. 807–817, ACM.

[7] Yilun Liu, Shimin Tao, Weibin Meng, Jingyu Wang, Wenbing Ma, Yuhang Chen, Yanqing Zhao, Hao Yang, and Yanfei Jiang, "Interpretable online log analysis using large language models with prompt strategies," in *Proceedings of the 32nd IEEE/ACM International Conference on Program Comprehension*. 2024, pp. 35–46, ACM.

[8] Wanhao Zhang, Qianli Zhang, Enyu Yu, Yuxiang Ren, Yeqing Meng, Mingxi Qiu, and Jilong Wang, "Lograg: Semi-supervised log-based anomaly detection with retrieval-augmented generation," in *IEEE International Conference on Web Services*. 2024, pp. 1100–1102, IEEE.

[9] Jonathan Pan, Swee Liang Wong, and Yidi Yuan, "Raglog: Log anomaly detection using retrieval augmented generation," *CoRR*, vol. abs/2311.05261, 2023.

[10] Xiao Han, Shuhan Yuan, and Mohamed Trabelsi, "Loggpt: Log anomaly detection via gpt," in *2023 IEEE International Conference on Big Data*, 2023, pp. 1117–1122.

[11] Sasho Nedelkoski, Jasmin Bogatinovski, Alexander Acker, Jorge Cardoso, and Odej Kao, "Self-attentive classification-based anomaly detection in unstructured logs," in *20th IEEE International Conference on Data Mining*. 2020, pp. 1196–1201, IEEE.

[12] Lin Yang, Junjie Chen, Zan Wang, Weijing Wang, Jiajun Jiang, Xuyuan Dong, and Wenbin Zhang, "Plelog: Semi-supervised log-based anomaly detection via probabilistic label estimation," in *ICSE Companion 2021*. 2021, pp. 230–231, IEEE.

[13] Zumin Wang, Jiyu Tian, Hui Fang, Liming Chen, and Jing Qin, "Lightlog: A lightweight temporal convolutional network for log anomaly detection on the edge," *Computer Networks*, vol. 203, pp. 108616, 2022.

[14] Tong Jia, Yifan Wu, Chuanjia Hou, and Ying Li, "Logflash: Real-time streaming anomaly detection and diagnosis from system logs for large-scale software systems," in *2021 IEEE 32nd ISSRE*, 2021, pp. 80–90.

[15] Xiaoda Xie, Songlei Jian, Chenlin Huang, Fengyuan Yu, and Yunjia Deng, "Logrep: Log-based anomaly detection by representing both semantic and numeric information in raw messages," in *2023 IEEE 34th ISSRE*, 2023, pp. 194–206.

[16] Xiaoyun Li, Pengfei Chen, Linxiao Jing, Zilong He, and Guangba Yu, "Swisslog: Robust and unified deep learning based log anomaly detection for diverse faults," in *31st IEEE International Symposium on Software Reliability Engineering*. 2020, pp. 92–103, IEEE.

[17] Crispin Almodovar, Fariza Sabrina, Sarvnaz Karimi, and Salahuddin Azad, "Logfit: Log anomaly detection using fine-tuned language models," *IEEE Transactions on Network and Service Management*, vol. 21, no. 2, pp. 1715–1723, 2024.

[18] Hongcheng Guo, Jian Yang, Jiaheng Liu, Jiaqi Bai, Boyang Wang, Zhoujun Li, Tieqiao Zheng, Bo Zhang, Junran Peng, and Qi Tian, "Logformer: A pre-train and tuning pipeline for log anomaly detection," in *Proceedings of the AAAI conference on artificial intelligence*, 2024, vol. 38, pp. 135–143.

[19] Song Chen and Hai Liao, "Bert-log: Anomaly detection for system logs based on pre-trained language model," *Applied Artificial Intelligence*, vol. 36, no. 1, pp. 2145642, 2022.

[20] Yukyung Lee, Jina Kim, and Pilsung Kang, "Lanobert: System log anomaly detection based on bert masked language model," *Applied Soft Computing*, vol. 146, pp. 110689, 2023.

[21] Chiming Duan, Tong Jia, Yong Yang, Guiyang Liu, Jinbu Liu, Huxing Zhang, Qi Zhou, Ying Li, and Gang Huang, "Eagerlog: Active learning enhanced retrieval augmented generation for log-based anomaly detection," in *ICASSP 2025 - 2025 IEEE ICASSP*, 2025, pp. 1–5.

[22] Yilun Liu, Yuhe Ji, Shimin Tao, Minggui He, Weibin Meng, Shenglin Zhang, Yongqian Sun, Yuming Xie, Boxing Chen, and Hao Yang, "Loglm: From task-based to instruction-based automated log analysis," in *2025 IEEE/ACM 47th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 2025, pp. 401–412.

[23] Yuhe Ji, Yilun Liu, Feiyu Yao, Minggui He, Shimin Tao, Xiaofeng Zhao, Chang Su, Xinhua Yang, Weibin Meng, Yuming Xie, Boxing Chen, and Hao Yang, "Adapting large language models to log analysis with interpretable domain knowledge," *CoRR*, vol. abs/2412.01377, 2024.

[24] Shilin He, Jieming Zhu, Pinjia He, and Michael R. Lyu, "Loghub: A large collection of system log datasets towards automated log analytics," *CoRR*, vol. abs/2008.06448, 2020.

[25] OpenAI, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, et al., "Gpt-4o system card," 2024.

[26] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo, "Deepseekmath: Pushing the limits of mathematical reasoning in open language models," 2024.

[27] Van-Hoang Le and Hongyu Zhang, "Log-based anomaly detection with deep learning: How far are we?," *CoRR*, vol. abs/2202.04301, 2022.

[28] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, et al., "Llama 2: Open foundation and fine-tuned chat models," 2023.