KAIOPS: A Platform Solution of End-to-End Multi-Modal AIOps for AI Training at Scale

Zeying Wang[1], Junhong Liu[1], Penghao Zhang[2], Xiaoyang Sun[32], Xu Wang[1], Tianyu Wo[1], Chunming Hu[1], Chengru Song[2], Jin Ouyang[2], Renyu Yang[1†]
[1]Beihang University [2]Kuaishou Inc. [3]University of Leeds

***Abstract*—The resilience of large-scale AI training platforms are fundamental to enabling contemporary AI innovation and business development. However, with the rapid increase in the scale and complexity of AI model training tasks, anomalies become the norm rather than the exception at scale. Failing to handle them properly may lead to enormous resource waste and prolonged development cycles. Traditional anomaly detection methods struggle to tackle the complex temporal characteristics and extreme class imbalance inherently manifesting in training tasks, and fall short in automated solution to root cause analysis and the follow-up remediation. This paper proposes KAIOPS, an end-to-end automated platform solution for handling anomalies and engineering experience of daily operational maintenance for large-scale AI training clusters at Kuaishou. KAIOPS employs a Temporal Context Encoding mechanism to precisely capture and encode long-term trends and critical temporal context information within fault evolution. The detection model elaborates a dynamic class-weighted loss function for enhancing the detection performance. To deliver a complete end-to-end intelligent processing pipeline, KAIOPS further leverages knowledge graph and LLMs for automated root cause analysis and actionable solution generation. Extensive experiments, on the basis of data collected from Kuaishou's production-grade training clusters, show the superior performance of our proposed approach. KAIOPS has been deployed in Kuaishou, in both testbed and production grade environments, consisting of with over 10,000 GPUs, and accelerate the reliability assurance for industry-scale model training and serving.**

***Index Terms*—AI Training Task, Anomaly Detection, Root Cause Analysis**

I. INTRODUCTION

In response to the ever-increasing demands for large-scale training tasks, dedicated high-performance AI clusters have been deployed as critical infrastructures by global high-tech companies [1]. These clusters accommodate massive workloads, including generative models and deep learning recommendation models, etc., and substantially underpin a wide variety of AI-driven innovations. However, as such workloads evolve in both volume and complexity, the risk of runtime anomalies within the large-scale AI clusters rapidly increases and the subsequent failures are likely to cause a huge amount of resource wastage, development delay, and monetary cost increase [2]. Our empirical study on a production-grade cluster in Kuaishou – one of China's largest short-video platforms – indicates that the task-level failure rate can reach as high as 19.3%. To ensure cluster stability and reliable task execution, timely detection, localization, and mitigation of these anomalies has therefore become a major operational challenge.

Fig. 1 depicts the de facto solution of handling anomalies that is highly dependent upon manual intervention. When an error occurs, SRE engineers have to manually search through FAQs and documentation, and where needed, they escalate the unresolved issues to on-call personnel. The holistic procedure can take 30 mins-1 hour, or even longer in some intricate cases. Undoubtedly, doing so is extremely labor-intensive and cause deteriorated Mean Time To Recovery (MTTR).

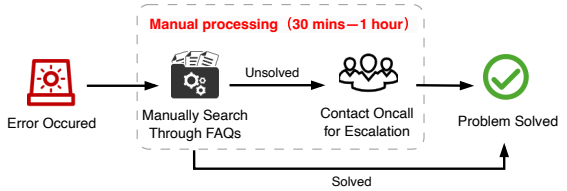


Fig. 1. An Example of Traditional Operation Process.

There are a body of anomaly detection approaches, encompassing deep learning models capturing complex correlations through end-to-end modeling (e.g., ART [3], Medicine [4]), Graph Neural Networks modelling structured dependencies within service call chains (e.g., DeepHunt [5], DiagFusion [6]), and knowledge-driven approaches integrating domain expertise and historical failure patterns (e.g., CloudRCA [7], MicroCBR [8]). They typically extract features from multi-source data (e.g., logs, monitoring metrics, and trace data) and employ multi-modal feature fusion for unified representation and joint analysis [9]. However, most of these approaches are largely exploited and fine-tuned for microservice systems. Tasks in AI infrastructures exhibit distinct characteristics – e.g., long runtime, high resource consumption, and intricate inter-node dependencies – that amplify the challenges of anomaly detection and expose huge limitations in applying existing methods to AI training at scale. This gap reveals an urgent need from both academia and industry.

Automated anomaly detection frameworks aiming to effectively manage anomalies in AI infrastructures face the following requirements[10]. *i) Observability Data.* A notable scarcity of standardized datasets hinders the implementation of effective detection approaches for training tasks. In fact, the characteristics of model training systems diverge significantly from those in traditional microservice systems. Domain-specific failure modes – such as GPU memory overflows, node disconnections, and configuration errors – require specialized feature extraction and modeling approaches[11, 12]. *ii) Subtle and Long-term Fault Patterns.* Unlike the immediate performance fluctuations of traditional systems, anomalies in training tasks often stem from gradual resource depletion, performance bottlenecks, or configuration errors. These nuanced and accumulative patterns challenge

†Corresponding author: Renyu Yang (renyuyang@buaa.edu.cn)

2643-1572/25/$31.00 ©2025 IEEE
DOI 10.1109/ASE63991.2025.00263

the existing methods' capability of achieving comprehensive detection[13–15]. *iii) Extreme Class Imbalance.* The frequency of different anomaly types varies drastically in real-world systems, leading to severe scarcity of long-tail samples. Such imbalance undermines the effectiveness and robustness of traditional supervised learning approaches[16]. Beyond anomaly detection and classification, production-grade AIOps solutions must deliver end-to-end anomaly management: engineering teams require explainable and automated systems capable of performing root cause analysis and generating actionable remediation to accelerate fault recovery.

In this paper, we propose KAIOPS, an end-to-end AIOps solution of anomaly management framework for large-scale AI training tasks. We first conduct in-depth analysis of training tasks to elaborate the multi-modal data model for better observability. KAIOPS identifies the faults at runtime through introducing temporal context encoding mechanism and defining dynamic class-weighted loss in the detection model. To identify complex anomaly patterns, the temporal context encoder integrates segmented convolutions with hybrid temporal attention – capturing long-term dependencies in fault evolution for effective anomaly detection. To tackle class imbalance, a novel dynamic class-weighted loss function is devised for enhancing the prediction accuracy for sparse fault types. We combine knowledge graph and Large Language Model to pinpoint the root causes and generate actionable repair policies. Specifically, KAIOPS integrates internal enterprise on-call historical data and FAQ documents to construct a specialized knowledge graph. A cloud-based LLM model is used for automated root cause analysis and solution provisioning. KAIOPS has been deployed in Kuaishou, in both testbed and production grade environments, to facilitate daily fault handling and operational management in production clusters consisting of over 10,000 GPUs. It has been used for reliability assurance for industry-scale model training and serving.

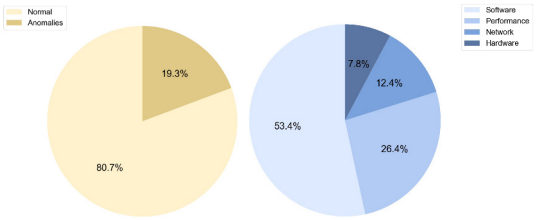
This paper makes the following contributions:

- The first multi-modal dataset, stemming from large-scale AI training tasks and encompassing a diverse range of typical fault types.
- A new detection framework, with the aid of temporal context encoding and dynamic class-weighted loss, to handle complex data schema, subtle fault patterns and class imbalance that are widely manifested in AI training tasks.
- An end-to-end AIOps solution for AI training tasks by leveraging the integration of knowledge graph and large language model, such that automated root cause analysis and solution recommendations can be fulfilled.

II. Background and Motivation

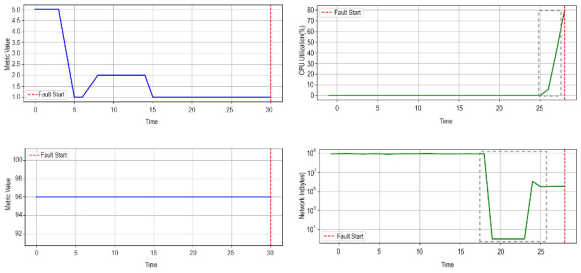
A. AI Model Training Clusters and Their Reliability

Large-scale AI training clusters have become key infrastructures and main driving force for industry players. Its reliability is pivotal to both infrastructure vendors and customers. However, the expansion of cluster scale and task complexity leads to a noticeable surge of anomalies during the training procedure. A failure may result in a non-negligible monetary cost, given that the whole training process has to be resumed from a checkpoint and a huge amount of computation resource and time will be wasted. Timely detecting and localizing the root causes therefore become ever-increasingly important yet challenging.



(a) Overall distribution. (b) Anomaly types distribution.

Fig. 2. The anomaly distribution in AI model training tasks.



(a) Metric curves in microservices. (b) Metric curves in AI clusters.

Fig. 3. Metric curves ahead of anomaly occurrence in different scenarios.

B. Anomaly Characteristics in Model Training Tasks

In industrial-level AI training scenarios, the distribution and patterns of anomalies fundamentally differ from those in cloud services such as microservices. We studied the anomalies manifested in Kuaishou's production-grade infrastructure, consisting of over 10,000 GPUs, within a few months observation period.

Observation-1: Diverse Anomaly Categories. As shown in Fig. 2, anomalies in industrial AI clusters can be categorized into four dinstinct yet interconnected types according to our engineering experience and real-world observations: i) *Software anomalies* typically include code errors, misconfigurations, and dependency conflicts, commonly occurring during task initialization or execution; ii) *Performance anomalies* are primarily manifest as irregular GPU utilization, Out-Of-Memory (OOM) , and computational performance degradation; iii) *Network anomalies* involve inter-node communication failures, bandwidth bottlenecks, and data transmission interruptions. iv) *Hardware anomalies* encompass GPU failures, storage device malfunctions, and node crashes. These anomalies can often trigger cascading failures, presenting significant challenges for fault diagnosis and remediation.

Observation-2: Class Imbalance. We also observed an extreme class imbalance in terms of the anomaly distribution. For example, software anomalies are relatively common, accounting for approximately 53.4%, while performance anomalies take up about 26.4%. Network and hardware anomalies are even rarer, accounting for only 12.4% and 7.8%, respectively. Such skewness makes it intricate to detect rare yet detrimental anomalies, such as hardware failures.

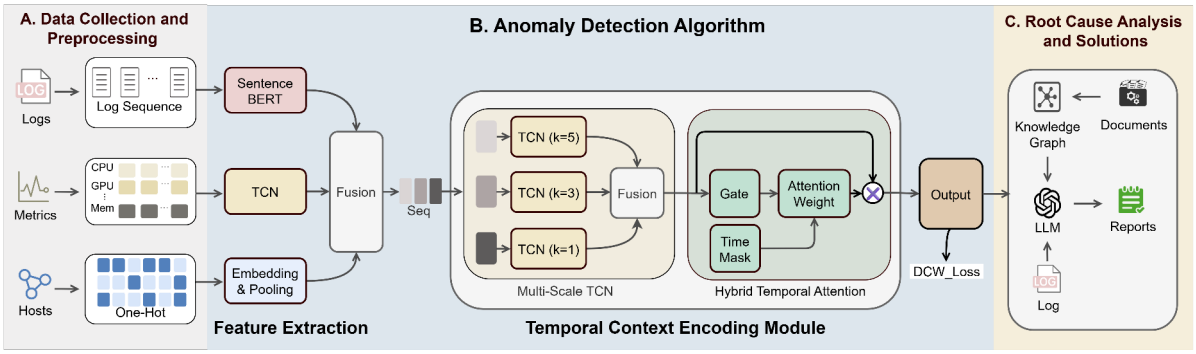


Fig. 4. The architecture of KAIOPS.

Traditional anomaly detection methods often perform poorly on minority classes due to insufficient training samples, resulting in high false positive or false negative rates.

Observation-3: Trending Anomaly Indicator. As opposed to anomalies in traditional microservice systems, often lacking persistent occurrence precursors, anomalies in AI training tasks manifest following subtle, long-term evolutionary trends. As illustrated in Fig. 3, anomalies in microservice systems(exemplified by two anomaly samples from the AIOps Challenge 2021 dataset[17]) often lack clear and persistent signs before they occur, while anomalies within training clusters frequently exhibit indicating evolutionary patterns before failure occurs. For instance, prior to an OOM failure, the memory utilization curves often show a spiking growth. Similarly, network throughput prior to communication failures tends to have escalating fluctuations or declines. Such trending-like shifts can serve as indicators of early failures, implying that data points temporally in close proximity to the failure event contain insightful warning signals.

C. Limitations of Existing Approaches

Anomaly handling in AI training clusters still heavily involves a huge amount of human intervention. When issues arise in training tasks, engineers typically need to log into the faulty nodes, manually inspect log files, monitoring metrics and system status, and determine the fault type and infer the pertaining root causes, and formulate remediation strategies. Manual diagnosis process has significant limitations:

- *Low efficiency and excessive MTTR.* Fault diagnosis is entirely dependent on operators' professional experience and domain knowledge, making it inefficient and time-consuming. Taking our on-site analysis of Kuaishou's large-scale training cluster as an example, the entire process of handling training task anomalies in the production environment, from anomaly occurrence to ultimate resolution, takes approximately 30 mins-1 hour on average. When dealing with the massive multi-modal data generated by large-scale clusters, manual analysis is highly inefficient, leading to a significant increase in MTTR and severely impacting business iteration velocity and the utilization of valuable computational resources.
- *High reliance on expert experience and poor scalability.* The accuracy of fault diagnosis is highly dependent on engineers' professional experience and domain knowledge. Novice operators often struggle to quickly pinpoint the root cause, potentially leading to further extensions in problem resolution time and even requiring cross-departmental communication and collaboration. This model is also difficult to scale when coping with the ever-increasing cluster size and fault numbers or types.
- *Limited diagnostic accuracy and coverage.* For complex anomaly patterns or rare fault types, manual diagnosis offers lower accuracy, prone to misdiagnosis or missed diagnoses. Furthermore, manual analysis often overlooks critical anomaly metrics at key timestamps and their interconnections, making it difficult to comprehensively capture the subtle and long-term fault patterns within training tasks.

III. OUR APPROACH

To address the limitations of existing anomaly detection methods in the context of AI training tasks, we propose KAIOPS – an end-to-end intelligent AIOps framework, as an aid of daily KAIOPS employs a three-stage architecture, spanning the entire workflow from data collection to automated solution generation. It is designed to intelligently and efficiently address anomalies in training tasks, thereby significantly enhancing operational efficiency. The overall architecture of KAIOPS is depicted in Fig. 4.

A. Data Collection and Preprocessing

The complex and multi-source nature of anomaly patterns in large-scale AI training clusters often renders traditional, single-dimensional monitoring data insufficient for comprehensive problem capture. To build a system capable of accurately identifying and localizing complex anomalies within training tasks, we conducted an extensive data collection effort on Kuaishou's production training platform, spanning a four-month period. This dataset encompasses over tens of thousands of training tasks, with their anomaly types meticulously labeled by a team of SRE experts.

We primarily collected the following three categories of critical monitoring data:

- **Logs:** Logs are detailed textual records generated during the execution of distributed training tasks. By aggregating logs from multiple machines and employing Log Parsing techniques, we convert unstructured logs into structured event sequences. These logs contain rich semantic information, revealing potential issues such as program errors, system events, and resource interactions.

- **Metrics:** We collected various critical performance metrics for each training task, specifically focusing on the 30-minute window preceding an anomaly event. These metrics include CPU utilization, memory consumption, GPU utilization, network throughput, and disk I/O. As minute-level time series data, these metrics reflect resource consumption and performance characteristics during task runtime.
- **Host Distribution Data:** This category represents a significant source of innovation for KAIOPS, as it captures the specific physical node allocation information for each training task. In distributed training environments, the topological structure of tasks and the co-location relationships among nodes are profoundly critical for system stability. However, traditional monitoring systems frequently overlook this crucial dimension. Our analysis revealed that distinct anomaly types exhibit clustering characteristics based on host distribution. For instance, as illustrated in Fig. 5, hardware anomalies typically concentrate within a specific subset of nodes. We collected this data by recording the host lists assigned by the task scheduler, integrating it as a vital "spatial" feature into our system.

B. Anomaly Detection Algorithm

The proposed anomaly detection algorithm is designed to effectively leverage the aforementioned multi-source heterogeneous data to efficiently identify potential anomalies within training clusters. Its core strength lies in its unique feature processing and fusion strategies, coupled with a meticulously designed loss function.

Multimodal Feature Extraction and Fusion. For log sequences, denoted as $\mathcal{L} = l_1, l_2, \ldots, l_T$ where l_i represents a single log entry in sequence, conventional approaches often employ word embedding techniques [18, 19] or BERT-based models [20] for feature extraction. However, these methods typically suffer from limitations such as insufficient semantic depth and high computational overhead. To achieve a balance between semantic processing depth and computational efficiency, we leverage a lightweight pre-trained Sentence-BERT model [21]. This model encodes each log sequence into a dense vector representation $\mathbf{L}' \in \mathbb{R}^{T \times d'_L}$, where T is the sequence length and $d'_L = 384$ is the embedding dimension. Subsequently, a multi-head self-attention mechanism is applied to model long-range dependencies within the log data. This process yields the final log feature representation $\mathbf{L} \in \mathbb{R}^{T \times d_L}$, with $d_L = 64$.

For metric sequences, denoted as $\mathcal{M} = m_1, m_2, \ldots, m_N$, each position within a sequence corresponds to a numerical feature vector. Each dimension of this vector represents a specific metric, such as CPU utilization, GPU utilization, or memory usage. Given their inherent time-series nature and the importance of local patterns, we employ a Temporal Convolutional Network (TCN) to encode the metric sequences. This results in the metric feature representation $\mathbf{M} \in \mathbb{R}^{T \times d_M}$, where $d_M = 32$.

Host data comprises the list of participating hosts within the distributed training environment. Each host $h_j \in \mathcal{H}$ (where $\mathcal{H}$ is the set of all available hosts) is vectorized into a $d_H = 32$ embedding. For a given training task, we extract the embeddings of all hosts involved in that specific task and apply a pooling operation to aggregate these host-specific embeddings into a unified, task-level host feature representation $\mathbf{H} \in \mathbb{R}^{d_H}$.

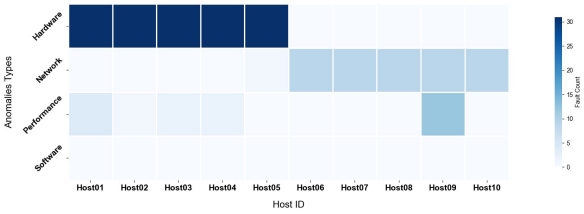


Fig. 5. Most frequently occurring host nodes in each anomaly category.

Finally, the extracted features from each modality—logs ($\mathbf{L}$), metrics ($\mathbf{M}$), and host information ($\mathbf{H}$)—are fused to form a comprehensive representation for each sample. The unified host feature vector $\mathbf{h}$ is broadcast and concatenated with the log and metric feature vectors at each time step $t \in 1, \ldots, T$. This results in the fused feature representation $\mathbf{X} \in \mathbb{R}^{T \times d_{fused}}$, where $d_{fused} = d_L + d_M + d_H$.

Temporal Context Encoding. Anomaly patterns in training tasks typically manifest as multi-scale temporal evolutions, often exhibiting an accelerating trend as they approach a failure point. To effectively capture these intricate dynamics and enable precise anomaly localization, we propose and design the temporal context encoding module, as depicted in Fig. 4. This module synergistically integrates multi-scale feature extraction, contextual dependency modeling, and dynamic attention focusing on critical information. This comprehensive approach yields robust feature representations that are highly sensitive to subtle anomaly manifestations.

The temporal context encoding module commences by processing the fused feature sequences via a novel multi-scale temporal convolution strategy. It partitions the sequence into distinct temporal segments (e.g., early, mid-term, recent), applying convolutional kernels with progressively smaller receptive fields to more recent segments. This design intuitively aligns with the notion that anomalies become more pronounced and localized closer to a failure event. The outputs from these segmented convolutions are then concatenated, forming a comprehensive feature representation that encapsulates diverse temporal granularities. Subsequently, to capture complex contextual dependencies within the time series, a Bidirectional Gated Recurrent Unit (Bi-GRU) is employed to process this concatenated representation. Building upon the critical observation that anomaly patterns significantly intensify and become more discernible as they approach a failure point, we further introduce an adaptive hybrid temporal attention mechanism. This mechanism dynamically modulates the model's focus across different timesteps and features within the Bi-GRU's output. Crucially, it is biased to amplify attention towards the later timesteps of the sequence, effectively prioritizing information from periods immediately preceding potential anomalies.

Dynamic Class-Weighted Loss Function. Class imbalance is a pervasive challenge in machine learning, often leading to models that underperform on minority classes [22]. Traditional solutions, such as static weighting [23] or resampling

[24], can introduce biases or fail to adapt to the evolving data distribution during training. To address these limitations, we propose a novel Dynamic Class-Weighted(DCW) Loss function, which integrates label smoothing and a dynamic class weighting mechanism to effectively handle class imbalance and improve model robustness.

Over-reliance on hard labels can lead to model overconfidence and reduced generalization, especially with noisy or ambiguous data. To mitigate this, we employ Label Smoothing, converting rigid one-hot encoded targets into softer probability distributions:

$$Soft\_Label = Onehot \cdot (1 - \delta) + \frac{\delta}{N}, \tag{1}$$

where $Onehot$ signifies the initial one-hot encoded vector for a given label, K is the number of categories, and δ is the smoothing factor.

To dynamically emphasize underperforming or underrepresented classes, we introduce a Dynamic Class Weighting strategy. Throughout training, we continuously monitor the accuracy for each class. Specifically, for each class n, we compute its accuracy:

$$Acc_n = C_n/(Num_n + \epsilon), \tag{2}$$

where C_n is the count of correct predictions and Num_n is the total count for class n, with ϵ being a small constant to prevent division by zero.

Based on these accuracies, we derive class-specific base weights:

$$W = 1.0/(Acc + \epsilon) \tag{3}$$

To modulate the sensitivity and distribution of these weights, we apply a temperature-scaled Softmax function:

$$W_{\text{softmax}} = \text{softmax}(W/T), \tag{4}$$

where a lower temperature T (e.g., $T < 1$) amplifies the weight differences, making the weighting more aggressive for poorly performing classes, while a higher T yields a smoother weight distribution.

We first calculate the standard cross-entropy loss for every sample in batch:

$$L_i = -\sum_{n=1}^{N} Soft\_Label_{i,n} \cdot \log(pred_{i,n}), \tag{5}$$

where L_i is the loss for sample i, and $pred_{i,n}$ is the predicted probability for sample i and class n.

We then apply the class-specific dynamic weights to each sample based on its true class:

$$L_{\text{weighted}} = \frac{1}{B}\sum_{i=1}^{B} L_i \cdot W_{\text{softmax}}[target_i], \tag{6}$$

where $W_{\text{softmax}}[target_i]$ is the dynamic weight corresponding to the true class of sample i.

This sample-level weighting mechanism ensures that samples from challenging categories receive higher weights during optimization, effectively channeling the model's learning capacity toward underperforming classes and improving classification performance on imbalanced datasets.

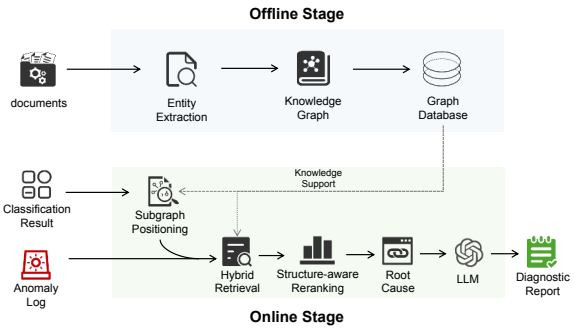


Fig. 6. Knowledge Graph-based Fault Diagnosis System Architecture.

C. Root Cause Analysis and Solutions

The knowledge graph construction module employs a hierarchical structured approach to automatically extract and organize fault diagnosis knowledge from internal enterprise on-call historical data and FAQ documents, and provide the root cause and solutions, the complete process diagram is shown in Fig. 6. The system adopts an offline-online dual-stage architecture design.

Offline Stage: The system first performs entity extraction on FAQ documents, converts them into semantic embedding vectors using Sentence-BERT models, constructs the knowledge graph and stores it in a graph database. The system creates four primary fault category nodes based on common anomaly types: software anomalies, network anomalies, performance anomalies, and hardware anomalies, serving as the top-level classification framework of the knowledge graph. For each category, a three-layer causal reasoning subgraph is constructed: *i) Symptom Layer*: stores specific fault phenomenon descriptions, such as task error "Resource exhausted: OOM when allocating tensor with shape *"; *ii) Root Cause Layer*: contains fundamental causes leading to symptoms, such as under KAI framework, large-scale models are prone to OOM, requiring different solutions based on memory OOM or GPU memory OOM scenarios; *iii) Solution Layer*: provides targeted remediation strategies and operational procedures. The system establishes directed relationship edges: `Symptom -[:BELONGS_TO]-> FaultCategory`, `Symptom -[:CAUSED_BY]-> RootCause`, `RootCause -[:RESOLVED_BY]-> Solution`. Additionally, technical term nodes such as "NCCL", "GPU", and "Graphics memory" are extracted through regular expressions and predefined terminology dictionaries, establishing `RELATED_TO` relationships with related entities to form a semantically rich heterogeneous knowledge network.

Online Stage: Root cause analysis and solution generation employs a four-stage pipeline architecture: *i) Subgraph Positioning Stage*: locates the target subgraph from the graph database based on classification results and anomaly logs; *ii) Hybrid Retrieval Stage*: executes keyword matching and semantic vector retrieval in parallel within the determined subgraph, where keyword retrieval rapidly locates relevant nodes containing technical terms through Cypher queries, while semantic retrieval computes cosine similarity between

TABLE I
RAG PROMPT EXAMPLE FOR TRAINING CLUSTER FAULT DIAGNOSIS

Prompt	Output
[System Role] You are an AI training cluster fault diagnosis expert. Provide analysis and solutions based on retrieved knowledge. **[Query]** Task error: "Nccl fail to connect." **[Retrieved Knowledge]** **Root Cause:** NCCL communication failure in distributed training. **Solutions:** Check network connectivity, verify NCCL/CUDA compatibility, examine GPU memory usage. **[Task]** 1) Analyze the fault cause based on the information provided by the above knowledge. 2) Provide detailed solution steps and explain why these steps can effectively solve the problem.	**Fault Description:** NCCL communication failure in distributed training. **Root Cause:** Multi-GPU/multi-node communication breakdown. **Solutions:** 1) Test network connectivity with ping; 2) Verify NCCL-CUDA version compatibility; 3) Check GPU memory and process conflicts; 4) Validate hostname resolution and SSH configuration.

TABLE II
DETAILS OF TRAIN-TASK-OPS DATASET

Dataset	Size	Labels	Counts
Train-Task-Ops	10,592	Normal	8,551
		Software Anomalies	1,089
		Performance Anomalies	537
		Network Anomalies	252
		Hardware Anomalies	163

anomaly descriptions and candidate node embedding vectors; *iii) Structure-aware Reranking Stage*: constructs NetworkX subgraphs and computes node degree centrality and betweenness centrality, employing the weighted fusion strategy $S_{final} = \alpha \cdot S_{semantic} + (1 - \alpha) \cdot S_{structure}$ to reorder retrieval results, where α balances semantic relevance and structural importance; *iv) Retrieval-augmented Generation Stage*: traverses knowledge graph relationships based on Top-1 root cause nodes to extract corresponding solution nodes and related technical information, inputs structured root cause analysis results and solutions into large language models through templated prompts, as the case shown in Table I, generating comprehensive diagnostic reports containing fault descriptions, environmental information, problem causes, and detailed solution steps.

IV. EVALUATION

This section systematically evaluates the performance of KAIOPS within practical industrial settings through a series of empirical experiments and analyses. We designed and conducted these experiments to address the following key research questions:

- **RQ1(Performance):** What is the overall performance of KAIOPS in anomaly detection for industrial-scale AI training tasks?
- **RQ2(Robustness):** How robust is KAIOPS in industrial scenarios characterized by extreme class imbalance?
- **RQ3(Data-wise Individual Contribution):** What is the individual contribution of different data modalities within KAIOPS to the overall detection performance?

A. Experiment Setup

Hardware and Software. We implement KAIOPS with Python 3.10 and the PyTorch deep learning framework. The experiments were conducted on a server equipped with two NVIDIA Tesla T4 GPUs (16GB memory each, CUDA 11.4 drivers) and an Intel Xeon Gold 6230R processor (x86_64 architecture, 2.1GHz base frequency) running Linux operating system. Key dependencies include Transformers 4.18.0 for pre-trained language models, Sentence-Transformers 2.2.2 for semantic embedding extraction, scikit-learn for baseline machine learning models, and standard libraries NumPy and Pandas for data processing.

Dataset. Addressing the current scarcity of publicly available real-world datasets for training task anomaly detection, our study constructed and leveraged a unique, proprietary dataset collected from Kuaishou's internal industrial-grade training platform. The entire process of data monitoring, collection, and organization was meticulously performed within Kuaishou's production training platform environment, thereby ensuring its authenticity and representativeness. Notably, all anomaly types were professionally annotated by a team of experienced SRE experts, which significantly enhances the dataset's accuracy and practical usability. The resulting dataset, named Train-Task-Ops, comprises diverse monitoring data and corresponding ground-truth labels for 10,592 distinct training tasks. This comprehensive collection provides a robust data foundation for our research. Detailed statistics and distribution of the dataset are presented in Table II.

Baselines. We compare KAIOPS with traditional machine learning methods, including decision trees and random forests, and a number of advanced deep learning approaches tailored for anomaly detection. For generalization, several comparative experiments are conducted on multi-source fusion data for anomaly detection.

- **Decision Tree** [25]: A traditional machine learning method that classifies data by constructing a decision tree structure.
- **Random Forest** [26]: An ensemble approach that combines multiple decision trees to improve accuracy and handle complex feature spaces through majority voting.
- **CloudRCA** [7]: A root cause analysis framework using diverse data sources and hierarchical Bayesian networks for efficient problem identification in cloud systems.
- **SCWarn** [27]: An approach for identifying bad software changes based on multimodal learning to accurately and timely identify bad changes and produce interpretable alerts.
- **Diagfusion** [6]: A failure diagnosis method combining metrics, logs, and traces with dependency graphs to accurately locate root causes and determine failure types.
- **Eadro** [28]: A framework integrating anomaly detection with root cause analysis for microservices, using multi-source data modeling and shared learning between phases.
- **Medicine** [4]: A diagnosis system that processes different monitoring data types independently while using adaptive optimization to balance their contributions, improving results even with incomplete data.

Metrics. For RQ1 and RQ3, we use Precision, Recall, and F1-score as performance evaluation metrics. In contrast, for

TABLE III
RESULTS OF DIFFERENT METHODS ON TRAIN-TASK-OPS DATASET

Method	Precision	Recall	F1-score
Decision Tree	0.6593	0.6242	0.6359
Random Forest	0.8051	0.6160	0.6568
CloudRCA	0.2815	0.2710	0.2738
Eadro	0.4054	0.4419	0.4223
Diagfusion	0.4216	0.4870	0.4392
SCWarn	0.6822	0.6514	0.6640
Medicine	0.7655	0.6637	0.6968
KAIOps	**0.7799**	**0.6947**	**0.7228**

RQ2, we use False Negative Rate (FNR) and False Positive Rate (FPR) to evaluate the model robustness.

B. Performance (RQ1)

To address Research Question RQ1 (RQ1), we systematically evaluated the performance of our proposed KAIOPS model on the real-world, industrial-scale Train-Task-Ops dataset. We rigorously compared its capabilities against a suite of state-of-the-art baseline methods, encompassing both traditional machine learning approaches and advanced deep learning frameworks.

Overall Performance Analysis. As shown in Table III, KAIOPS demonstrates superior performance across all evaluation metrics, achieving the highest Precision (0.7799), Recall (0.6947), and F1-score (0.7228). This initial validation unequivocally confirms KAIOPS's effectiveness in training task anomaly detection.

Detailed Performance and Baseline Comparison. Table IV further provides a fine-grained breakdown of performance across different methods in detecting various specific anomaly types (i.e., performance anomalies, network anomalies, hardware anomalies, software anomalies) and normal samples. Decision Tree [25] and Random Forest [26], as classic machine learning models, show stable overall performance but significant limitations with specific anomaly types, particularly low F1-scores for rare anomalies like network and hardware anomalies. CloudRCA [7] exhibited the poorest performance (F1-score: 0.2738), primarily because its design targets general-purpose cloud computing platforms, and its core mechanisms struggle to effectively transfer and adapt to the unique data patterns and anomaly characteristics inherent in training tasks. Eadro [28] (F1-score: 0.4223) and Diagfusion [6] (F1-score: 0.4392) achieved moderate overall performance. However, it is noteworthy that both completely failed to detect network and hardware anomalies, highlighting their lack of robustness when handling these specific anomaly types. SCWarn [27] performed commendably (F1-score: 0.6640), with its strength lying in its sensitivity to feature changes. Nevertheless, its limitation stems from an over-reliance on detecting discrete feature variations, often neglecting the broader evolution of data trends, which is crucial for comprehensively capturing anomaly patterns. Among all baseline methods, Medicine exhibited the best performance, particularly in performance anomaly detection (F1-score: 0.745). However, Medicine demonstrates a clear imbalance in its detection capabilities: it performs poorly in network anomaly detection (F1-score: 0.417) and shows significantly low Recall in hardware anomaly detection (Recall: 0.400). This indicates Medicine's difficulty in comprehensively and consistently identifying all types of training task anomalies.

Compared to all baseline methods, KAIOPS demonstrates superior and balanced multi-type anomaly detection capabilities. As detailed in Table IV, KAIOPS achieves optimal or near-optimal performance in detecting network anomalies (F1-score: 0.519), hardware anomalies (F1-score: 0.609), and software anomalies (F1-score: 0.800). Furthermore, KAIOPS maintains excellent performance in identifying normal samples (F1-score: 0.990), comparable to Medicine.

> **Answer to RQ1:** KAIOPS significantly outperforms the state-of-the-art baselines on the Train-Task-Ops dataset. Such superiority stems from the modal interaction mechanism and the tailored optimizations for the unique characteristics of training task scenarios.

C. Robustness (RQ2)

In real-world industrial deployments, False Negative Rate (FNR) and False Positive Rate (FPR) are pivotal metrics for assessing the robustness and practical utility of anomaly detection systems. A high FNR indicates the system's failure to detect genuine anomalies, which can lead to persistent critical failures and subsequent task disruptions. Conversely, a high FPR results in an excessive number of false alarms, leading to "alert fatigue" among associated personnel. Consequently, RQ2 aims to evaluate KAIOPS's capability to effectively balance and minimize these two crucial metrics within severely class-imbalanced industrial settings.

As presented in Table V, KAIOPS demonstrates exceptional robust performance concerning both FNR and FPR. KAIOPS achieved an FNR of 0.3053, representing an approximate 9.2% reduction compared to the best-performing baseline method, Medicine (0.3363). This substantiates KAIOPS's enhanced capability to capture rare, genuine anomaly events within industrial environments, thereby significantly mitigating the risk of missed fault detections. Concurrently, KAIOPS's FPR stands at merely 0.0202, representing the optimal value across all compared methods and significantly lower than most baseline approaches. This implies KAIOPS's substantial capacity to minimize false alarms, effectively alleviating alert fatigue and, consequently, providing reliable and actionable anomaly alerts in real operational settings.

These results demonstrate KAIOPS's capability to maintain precise and stable anomaly detection in class-imbalanced industrial scenarios.

> **Answer to RQ2:** KAIOPS demonstrates exceptional robustness in severely class-imbalanced industrial scenarios. It effectively balances false negatives and false positives by significantly reducing both the False Negative Rate (FNR) and False Positive Rate (FPR), thereby ensuring the accuracy and reliability of its detection results.

D. Data Source Contribution (RQ3)

To quantify the contribution of individual data modalities to KAIOPS's overall detection efficacy and to validate the

TABLE IV
RESULTS OF DIFFERENT METHODS ON TRAIN-TASK-OPS DATASET BY ANOMALY TYPES

Method	Performance Anomaly			Network Anomaly			Hardware Anomaly			Software Anomaly			Normal		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Decision Tree	0.670	0.589	0.627	0.444	0.400	0.421	0.524	0.344	0.415	0.672	0.811	0.735	0.986	0.977	0.982
Random Forest	0.696	0.729	0.712	0.647	0.220	0.328	**1.000**	0.344	0.512	0.703	0.797	0.747	0.979	0.990	0.984
CloudRCA	0.095	0.075	0.084	0.048	0.060	0.053	0.032	0.061	0.042	0.391	0.303	0.341	0.843	0.857	0.850
Diagfusion	0.414	**0.837**	0.554	0.000	0.000	0.000	0.000	0.000	0.000	0.710	0.606	0.654	0.984	0.993	0.989
Eadro	0.482	0.531	0.505	0.000	0.000	0.000	0.000	0.000	0.000	0.580	0.697	0.633	0.965	0.982	0.974
SCWarn	0.647	0.674	0.660	0.417	0.385	0.400	0.636	**0.467**	0.539	0.723	0.743	0.733	0.988	0.989	0.989
Medicine	**0.778**	0.714	**0.745**	0.455	0.385	0.417	0.857	0.400	0.546	0.750	0.826	0.786	0.988	**0.994**	**0.991**
KAIOps	0.775	0.633	0.697	**0.500**	**0.539**	**0.519**	0.875	**0.467**	**0.609**	**0.760**	**0.844**	**0.800**	**0.989**	0.992	0.990

TABLE V
FNR AND FPR RESULTS ON TRAIN-TASK-OPS DATASET

Method	FNR	FPR
Decision Tree	0.3758	0.0269
Random Forest	0.3840	0.0299
CloudRCA	0.7290	0.1644
Eadro	0.5581	0.0528
Diagfusion	0.5130	0.0309
SCWarn	0.3486	0.0241
Medicine	0.3363	0.0213
KAIOps	**0.3053**	**0.0202**

TABLE VI
ABLATION EXPERIMENT OF DIFFERENT MODALITIES OF DATA

Approach	Precision	Recall	F1-score
w/o H	0.6475	0.6144	0.6179
w/o M	0.7222	0.5988	0.6316
w/o L	0.7563	0.4122	0.4872
KAIOps	**0.7799**	**0.6947**	**0.7228**

effectiveness of multi-source heterogeneous data fusion, we conducted a series of ablation studies. In these experiments, we individually removed one data modality (i.e., Host Information (w/o H), Metrics (w/o M), and Logs (w/o L)) and evaluated KAIOPS's performance with the remaining data modality combinations. Table VI presents the Precision, Recall, and F1-score performance after the removal of each data modality, juxtaposed against the complete KAIOPS that incorporates all modalities.

Experimental results demonstrate that all data modalities are critical for KAIOPS's overall detection efficacy, exhibiting significant synergistic contributions. Logs emerged as a core driving force; removing log data caused the F1-score to plummet to 0.4872, representing the most substantial decrease of 0.2356 compared to the full KAIOPS's 0.7228. This highlights the irreplaceable nature of the rich semantic information contained within logs for identifying fault root causes and complex anomaly patterns. Its Recall experienced a particularly pronounced drop (from 0.6947 to 0.4122), underscoring the crucial role of logs in capturing the majority of true anomalies. Metrics provide essential performance context; when metric data was removed, the F1-score declined to 0.6316. This indicates that performance metrics offer vital information regarding resource consumption and performance bottlenecks, contributing to the timely detection of anomalies stemming from resource exhaustion or performance degradation. Host Distribution Data serves as a unique complementary element; upon its removal, the F1-score dropped to 0.6179, and its Precision saw a particularly sharp decline from 0.7799 to 0.6475. This suggests that Host Distribution Data, as a distinctive innovation of KAIOPS, is more than merely providing additional features; it functions as a crucial calibration or discriminatory signal. By revealing spatial characteristics and co-location relationships of tasks on physical nodes, and particularly potent for uniquely identifying issues related to hardware failures or specific node groups, host data enables the model to more accurately discern the presence of true anomalies and significantly reduces false positives.

To sum up, these findings indicate that KAIOPS achieves comprehensive anomaly detection by integrating semantic information from logs, performance context from metrics, and spatial features from host distribution data.

Answer to RQ3: All data modalities within KAIOPS are indispensable for its overall detection performance and demonstrate synergistic effects. Specifically, log data provides core semantic information for identifying fault patterns; the uniquely integrated host distribution data serves as a critical discriminatory signal; while metric data complements with crucial resource and performance context. This comprehensive fusion of multi-source heterogeneous data is pivotal for KAIOPS to achieve its superior anomaly detection performance and accurate discrimination.

V. CASE STUDIES AND EXPERIENCE IN KUAISHOU

KAIOPS has been deployed in Kuaishou, in both testbed and production grade environments, to facilitate daily anomaly handling and operational management in production clusters, each with over 10,000 GPUs, and serve industry-scale real-time video recommendation services.

A. Real-World Case Studies

To comprehensively validate KAIOPS's practical capabilities in addressing diverse anomalies and its efficiency in resolving them within AI training tasks, we dive into four representative real-world industrial scenarios.

TABLE VII
CASE STUDIES: TYPICAL ANOMALY MANAGEMENT USING KAIOPS

Anomaly Type	Observed Phenomenon	Diagnosed Root Causes	Recommended Solutions
Software	Task fails at startup with "Expected embedding weights to have size `[64, 1024]`, but got `[64, 512]` instead." error.	Embedding layer dimension mismatch due to incompatible model config/version.	Update model configuration parameters (e.g., embedding dim) to match expected dimensions; provide correct config example.
Performance	Training task terminates suddenly after hours; last log is "CUDA out of memory." No prior threshold alerts.	GPU memory leak/excessive consumption (e.g., overly large batch size, inefficient memory management) leading to OOM.	Reduce batch size; enable gradient accumulation; optimize model's memory footprint; check for explicit memory leaks in code.
Network	Distributed training slowdown, frequent "NCCL WARN" or "connection timed out" warnings. Intermittent	Specific network link/switch port bottleneck or intermittent fault affecting communication between specific nodes (identified by host distribution).	Check/replace specific switch port; inspect network cables; analyze traffic on identified network link for congestion/errors.
Hardware	Model loss function converges abnormally; subtle output deviations. Logs show "CUDA: uncorrectable ECC error encountered." No crash.	GPU hardware fault (e.g., uncorrectable memory error on a specific GPU) leading to silent data corruption.	Immediately isolate the faulty GPU/server; schedule hardware replacement to prevent further erroneous computations.

As shown in Table VII, we validated KAIOPS's exceptional effectiveness across various anomaly scenarios through concrete case studies. For software anomalies, such as task startup failures due to embedding layer dimension mismatch, KAIOPS leverages its deep understanding of log semantics and integration with a knowledge graph to swiftly pinpoint configuration inconsistencies as the root cause, providing precise remediation suggestions. When addressing performance anomalies, such as subtle GPU Memory Out-Of-Memory (OOM) issues, KAIOPS's temporal context encoding module accurately captures the nuanced trend of continuously accelerating GPU memory utilization before an OOM event, thus enabling early warning and proposing optimization strategies to effectively prevent task crashes and resource wastage. For prevalent network anomalies like NCCL connection issues, KAIOPS integrates logs, metrics, and its uniquely incorporated host distribution data to uncover physical-layer network link bottlenecks, precisely pinpointing the root cause and providing targeted recommendations for network device inspection. Even for subtle hardware anomalies, such as silent GPU computation errors like "CUDA: uncorrectable ECC error", KAIOPS can detect them from minimal log entries even without explicit system crashes, promptly localize the faulty GPU, and recommend isolation and replacement.

This end-to-end intelligent diagnosis and resolution process, from anomaly occurrence to solution provision, takes less than 30 seconds on average. This drastically improves the operational efficiency and reliability of AI training platforms compared to traditional manual troubleshooting processes that heavily rely on human expertise and can take several hours. These case studies not only highlight KAIOPS's proficiency in identifying complex, subtle, and distributed anomalies but also underscore its end-to-end capability, spanning from intelligent detection to root cause analysis and solution generation.

B. Engineering Experience

Guaranteeing Model Training Stability. In large-scale AI training clusters – where jobs run for days/weeks across thousands of accelerators –runtime anomaly detection and diagnosis acts as a mission-critical safeguard. It is highly desirable to devise effective and accurate detection methods that can quickly react to software and hardware faults at runtime. This need necessitates continuously monitoring training workloads during execution, identifying deviations from expected behavior and pinpointing underlying causes before failures escalate.

KAIOPS is deployed to harness different kinds of anomalies and enhances the training stability. Specifically, KAIOPS is already proved effective in early-stage prevention, e.g., detecting subtle anomalies in real-time (e.g., memory leaks, network congestion, skewed gradients) before they trigger job crashes. The generation of actionable fixes, with the aid of LLMs and knowledge graphs, also enables the proactive intervention. For example, the NCCL fail-slow such as 30% spike in NCCL timing-out could help the associated engineering team to redirect traffic or performing better request orchestration to bypass the faulty switch devices. It is also beneficial for optimizing resource allocation and job-level reconfiguration. The detected stragglers could be temporarily removed or substituted by restarting corrupted processes. The raw telemetry data such as metrics, logs, traces can be transformed into insightful reconfiguration during runtime, helping the model designer to tweak the most suitable parameters and make the best use of the hardware accelerators.

Applicability and Generality. While KAIOPS is coupled with Kuaishou's distributed model training platform KAI and monitoring platform, one can easily reproduce the end-to-end solution based on individual AI training infrastructures. This can be simply achieved by adapting the multimodal data interfaces, i.e., using a unified observability stardard APIs, and by customizing their own representation learning and anomaly detection algorithm.

We are aware that not all system-level anomalies or performance degradation are captured by KAIOPS, particularly considering the extra-scale model training. The KAIOPS's capability of harnessing anomalies can be massively enriched by integrating other monitoring tools for fine-grained extensible ebpf-based tracing, RDMA traffic diagnosis, etc. The newly tracked and collected multimodal data, covering computation, storage, and communication aspects, can be

further used for online training and parameter update [29].

VI. Related work

A. Single-Modal Anomaly Detection

Log-based anomaly detection methods have evolved from simple count indices to semantic understanding. Early approaches using traditional machine learning primarily employed log count vectors [30–32]. Subsequently, LogAnomaly[30] and SwissLog[31] model log streams as natural language sequences to extract semantic information. Advanced methods such as VCRLog[32] mine relationships among discrete variables, and semi-supervised approaches like LogOnline[33], AFALog[34] and LogCAE[35] incorporate online learning mechanisms. However, these single-modal log-based approaches struggle with the complex temporal dynamics and extreme class imbalance inherent in AI training tasks, often failing to capture the multi-faceted nature of training anomalies that require comprehensive contextual understanding across multiple data modalities.

Metric-based anomaly detection methods demonstrate enhanced interpretability and decision support capabilities. DéjàVu[36] and iSQUAD[37] focus on interpretable anomaly diagnosis for recurring failures and intermittent slow queries. Deep learning approaches like ImDiffusion[38] combine time series imputation and diffusion models to capture temporal dependencies. In adaptive diagnosis, MS-Rank[39] and MicroCause[40] implement dynamic weight updates and causal inference. For real-time operation and automation, VersaGuardian[41] introduces dynamic mode decomposition technology, while AutoKAD[42] addresses algorithm selection and hyperparameter optimization through label-free universal objective functions and cluster-based warm start strategies. Nevertheless, existing methods primarily target traditional IT infrastructure anomalies and lack specialized temporal context encoding mechanisms needed for AI workloads, where fault evolution patterns exhibit unique characteristics requiring domain-specific modeling approaches.

Trace-based anomaly detection methods emphasize system-level analysis and root cause localization. TraceRCA[43] conducts root cause localization based on the ratio of abnormal to normal traces, while MicroHECL[44] improves accuracy through dynamic service call graphs and anomaly propagation analysis. MEPFL[45] trains predictive models at trace and microservice levels for latent error prediction. For end-to-end latency localization, MicroRank[46] employs PageRank scoring for trace importance, while MicroSketch[47] adopts lightweight Sketch feature analysis. Additionally, TraceAnomaly[48] and GMTA[49] integrate graph analysis and deep learning technologies, enhancing detection accuracy through service-level deep Bayesian networks and graph representation learning. However, these approaches primarily target microservices architectures, and lack capability to handle unique execution patterns and resource utilization characteristics of AI training tasks.

B. Multi-Modal Anomaly Detection

Multi-modal data anomaly detection is advanced by the fact that a single data source cannot comprehensively characterize complex system behaviors. Early works like CloudRCA[7] and PDiagnose[50] demonstrated significant advantages of multi-source integration for root cause analysis. Subsequent works fall into three aspects. First, end-to-end frameworks emerged, with Eadro[28] integrating anomaly detection and root cause localization, and ART[3] unifying detection, triage, and localization, validating the value of multi-task learning. While Medicine[4] and Diagfusion[27] addressed modal balance and data imbalance issues through modality-independent frameworks and embedding augmentation. Second, graph-based methods including DeepHunt[5], MULAN[51] and AnoFusion[52] leveraged graph structures to capture complex relationships through autoencoders, causal learning, and transformer networks. Third, knowledge-driven and case-based reasoning introduced new perspectives, with MicroCBR[8], SCELM[53] and Nezha[54] incorporating spatio-temporal knowledge graphs and large language models for interpretable analysis. Additionally, SCWarn[27] provided interpretable change alerts through multi-modal learning, making fault diagnosis results more actionable.

However, existing multimodal approaches primarily target general IT infrastructure, falling short in handling unique challenges inherent in AI training workloads: extreme class imbalance, complex temporal evolution patterns across training cycles, and the need for domain-specific knowledge integration to provide actionable end-to-end solutions for enterprise cluster management and training optimization.

VII. Conclusion

This work proposes KAIOps, a novel end-to-end AIOps framework specifically designed to address the complex resilience and reliability challenges in large-scale AI training. By constructing a multi-modal analysis pipeline that innovatively integrates temporal context encoding, dynamic class-weighted loss, and knowledge-driven Large Language Model inference capabilities, KAIOps comprehensively tackles the unique complexities of AI training task anomalies—ranging from their subtle temporal evolution and extreme class imbalance issues to the pressing demand for automated diagnosis. Experiments and case-studies show exceptional detection accuracy and practical management efficacy. This industrial-grade practice provides a valuable blueprint for building automated AIOps capabilities for AI training platforms.

Acknowledgment

We would very much like to thank the anonymous reviewers for their valuable comments. Special thanks must go to the overall AI platform team at Kuaishou Inc. and RAIDS Lab team at Beihang University, for their constant support, collaborative contribution and countless technical discussion.

This work is supported in part by National Key R&D Program of China (Grant No. 2024YFB4505901), in part by the National Natural Science Foundation of China (Grant No. 62402024), in part by the Beijing Natural Science Foundation (Grant No. L241050), in part by the Fundamental Research Funds for the Central Universities, and, last but not the least, by Kuaishou Research Fund. For any correspondence, please refer to the project lead and coordinator Dr. Renyu Yang (renyuyang@buaa.edu.cn).

REFERENCES

[1] H. Wang, Z. Qu, Q. Zhou, H. Zhang, B. Luo, W. Xu, S. Guo, and R. Li, "A comprehensive survey on training acceleration for large machine learning models in iot," *IEEE Internet of Things Journal*, vol. 9, no. 2, pp. 939–963, 2021.

[2] A. Kokolis, M. Kuchnik, J. Hoffman, A. Kumar, P. Malani, F. Ma, Z. DeVito, S. Sengupta, K. Saladi, and C.-J. Wu, "Revisiting reliability in large-scale machine learning research clusters," in *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2025, pp. 1259–1274.

[3] Y. Sun, B. Shi, M. Mao, M. Ma, S. Xia, S. Zhang, and D. Pei, "Art: A unified unsupervised framework for incident management in microservice systems," in *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, 2024, pp. 1183–1194.

[4] L. Tao, S. Zhang, Z. Jia, J. Sun, M. Ma, Z. Li, Y. Sun, C. Yang, Y. Zhang, and D. Pei, "Giving every modality a voice in microservice failure diagnosis via multimodal adaptive optimization," in *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, 2024, pp. 1107–1119.

[5] Y. Sun, Z. Lin, B. Shi, S. Zhang, S. Ma, P. Jin, Z. Zhong, L. Pan, Y. Guo, and D. Pei, "Interpretable failure localization for microservice systems based on graph autoencoder," *ACM Transactions on Software Engineering and Methodology*, vol. 34, no. 2, pp. 1–28, 2025.

[6] S. Zhang, P. Jin, Z. Lin, Y. Sun, B. Zhang, S. Xia, Z. Li, Z. Zhong, M. Ma, W. Jin *et al.*, "Robust failure diagnosis of microservice system through multimodal data," *IEEE Transactions on Services Computing*, vol. 16, no. 6, pp. 3851–3864, 2023.

[7] Y. Zhang, Z. Guan, H. Qian, L. Xu, H. Liu, Q. Wen, L. Sun, J. Jiang, L. Fan, and M. Ke, "Cloudrca: A root cause analysis framework for cloud computing platforms," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 4373–4382.

[8] F. Liu, Y. Wang, Z. Li, R. Ren, H. Guan, X. Yu, X. Chen, and G. Xie, "Microcbr: Case-based reasoning on spatio-temporal fault knowledge graph for microservices troubleshooting," in *International Conference on Case-Based Reasoning*. Springer, 2022, pp. 224–239.

[9] P. Xu, Q. Gao, Z. Zhang, and K. Zhao, "Multi-source data based anomaly detection through temporal and spatial characteristics," *Expert Systems with Applications*, vol. 237, p. 121675, 2024.

[10] L. Myllyaho, M. Raatikainen, T. Männistö, T. Mikkonen, and J. K. Nurminen, "Systematic literature review of validation methods for ai systems," *arXiv preprint arXiv:2107.12190*, 2021.

[11] Y. He, M. Hutton, S. Chan, R. De Gruijl, R. Govindaraju, N. Patil, and Y. Li, "Understanding and mitigating hardware failures in deep learning training systems," in *Proceedings of the 50th Annual International Symposium on Computer Architecture*, 2023, pp. 1–16.

[12] Z. Zhang, L. Huang, R. Huang, W. Xu, and D. S. Katz, "Quantifying the impact of memory errors in deep learning," in *2019 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2019, pp. 1–12.

[13] H. V. Pham, S. Qian, J. Wang, T. Lutellier, J. Rosenthal, L. Tan, Y. Yu, and N. Nagappan, "Problems and opportunities in training deep learning software systems: An analysis of variance," in *Proceedings of the 35th IEEE/ACM international conference on automated software engineering*, 2020, pp. 771–783.

[14] K. Maeng, S. Bharuka, I. Gao, M. Jeffrey, V. Saraph, B.-Y. Su, C. Trippel, J. Yang, M. Rabbat, B. Lucia *et al.*, "Understanding and improving failure tolerant training for deep learning recommendation with partial recovery," *Proceedings of Machine Learning and Systems*, vol. 3, pp. 637–651, 2021.

[15] T. Liao, R. Taori, I. D. Raji, and L. Schmidt, "Are we learning yet? a meta review of evaluation failures across machine learning," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[16] R. Zhang, W. Xiao, H. Zhang, Y. Liu, H. Lin, and M. Yang, "An empirical study on program failures of deep learning jobs," in *Proceedings of the ACM/IEEE 42nd international conference on software engineering*, 2020, pp. 1159–1170.

[17] https://www.aiops.cn/gitlab/aiops-nankai/data/trace/aiops2021.

[18] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[19] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, 2013.

[20] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, 2019, pp. 4171–4186.

[21] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv preprint arXiv:1908.10084*, 2019.

[22] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on knowledge and data engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[23] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[24] A. Estabrooks, T. Jo, and N. Japkowicz, "A multiple resampling method for learning from imbalanced data sets," *Computational intelligence*, vol. 20, no. 1, pp. 18–36, 2004.

[25] L. Ying *et al.*, "Decision tree methods: applications for classification and prediction," *Shanghai archives of psychiatry*, vol. 27, no. 2, p. 130, 2015.

[26] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.

[27] N. Zhao, J. Chen, Z. Yu, H. Wang, J. Li, B. Qiu, H. Xu, W. Zhang, K. Sui, and D. Pei, "Identifying bad software changes via multimodal anomaly detection for online service systems," in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2021, pp. 527–539.

[28] C. Lee, T. Yang, Z. Chen, Y. Su, and M. R. Lyu, "Eadro: An end-to-end troubleshooting framework for microservices on multi-source data," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023, pp. 1750–1762.

[29] Z. Liu, R. Yang, J. Ouyang, W. Jiang, T. Ye, M. Zhang, S. Huang, J. Huang, C. Song, D. Zhang *et al.*, "Kale: Elastic gpu scheduling for online dl model training," in *Proceedings of the 2024 ACM Symposium on Cloud Computing*, 2024, pp. 36–51.

[30] W. Meng, Y. Liu, Y. Zhu, S. Zhang, D. Pei, Y. Liu, Y. Chen, R. Zhang, S. Tao, P. Sun *et al.*, "Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs." in *IJCAI*, vol. 19, no. 7, 2019, pp. 4739–4745.

[31] X. Li, P. Chen, L. Jing, Z. He, and G. Yu, "Swisslog: Robust anomaly detection and localization for interleaved unstructured logs," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 4, pp. 2762–2780, 2022.

[32] J. Wang, T. Li, R. Zhang, Z. Tang, D. Wu, and Z. Yang, "Vcrlog: Variable contents relationship perception for log-based anomaly detection," in *2024 IEEE 35th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2024, pp. 156–167.

[33] X. Wang, J. Song, X. Zhang, J. Tang, W. Gao, and Q. Lin, "Lo-

gonline: A semi-supervised log-based anomaly detector aided with online learning mechanism," in *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2023, pp. 141–152.

[34] C. Duan, T. Jia, H. Cai, Y. Li, and G. Huang, "Afalog: A general augmentation framework for log-based anomaly detection with active learning," in *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2023, pp. 46–56.

[35] P. Xiao, T. Jia, C. Duan, H. Cai, Y. Li, and G. Huang, "Logcae: An approach for log-based anomaly detection with active learning and contrastive learning," in *2024 IEEE 35th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2024, pp. 144–155.

[36] Z. Li, N. Zhao, M. Li, X. Lu, L. Wang, D. Chang, X. Nie, L. Cao, W. Zhang, K. Sui *et al.*, "Actionable and interpretable fault localization for recurring failures in online service systems," in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2022, pp. 996–1008.

[37] M. Ma, Z. Yin, S. Zhang, S. Wang, C. Zheng, X. Jiang, H. Hu, C. Luo, Y. Li, N. Qiu *et al.*, "Diagnosing root causes of intermittent slow queries in cloud databases," *Proceedings of the VLDB Endowment*, vol. 13, no. 8, pp. 1176–1189, 2020.

[38] Y. Chen, C. Zhang, M. Ma, Y. Liu, R. Ding, B. Li, S. He, S. Rajmohan, Q. Lin, and D. Zhang, "Imdiffusion: Imputed diffusion models for multivariate time series anomaly detection," *arXiv preprint arXiv:2307.00754*, 2023.

[39] M. Ma, W. Lin, D. Pan, and P. Wang, "Self-adaptive root cause diagnosis for large-scale microservice architecture," *IEEE Transactions on Services Computing*, vol. 15, no. 3, pp. 1399–1410, 2020.

[40] Y. Meng, S. Zhang, Y. Sun, R. Zhang, Z. Hu, Y. Zhang, C. Jia, Z. Wang, and D. Pei, "Localizing failure root causes in a microservice through causality inference," in *2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS)*. IEEE, 2020, pp. 1–10.

[41] L. Tao, M. Ma, S. Zhang, J. Kuang, X.-W. Guo, C. Yang, and D. Pei, "Real-time anomaly detection for large-scale network devices," *IEEE Transactions on Networking*, 2025.

[42] Z. Yu, C. Pei, S. Zhang, X. Wen, J. Li, G. Xie, and D. Pei, "Autokad: Empowering kpi anomaly detection with label-free deployment," in *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2023, pp. 13–23.

[43] Z. Li, J. Chen, R. Jiao, N. Zhao, Z. Wang, S. Zhang, Y. Wu, L. Jiang, L. Yan, Z. Wang *et al.*, "Practical root cause localization for microservice systems via trace analysis," in *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*. IEEE, 2021, pp. 1–10.

[44] D. Liu, C. He, X. Peng, F. Lin, C. Zhang, S. Gong, Z. Li, J. Ou, and Z. Wu, "Microhecl: High-efficient root cause localization in large-scale microservice systems," in *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 2021, pp. 338–347.

[45] X. Zhou, X. Peng, T. Xie, J. Sun, C. Ji, D. Liu, Q. Xiang, and C. He, "Latent error prediction and fault localization for microservice applications by learning from system trace logs," in *Proceedings of the 2019 27th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering*, 2019, pp. 683–694.

[46] G. Yu, P. Chen, H. Chen, Z. Guan, Z. Huang, L. Jing, T. Weng, X. Sun, and X. Li, "Microrank: End-to-end latency issue localization with extended spectrum analysis in microservice environments," in *Proceedings of the Web Conference 2021*, 2021, pp. 3087–3098.

[47] Y. Li, G. Yu, P. Chen, C. Zhang, and Z. Zheng, "Microsketch: Lightweight and adaptive sketch based performance issue detection and localization in microservice systems," in *International Conference on Service-Oriented Computing*. Springer, 2022, pp. 219–236.

[48] P. Liu, H. Xu, Q. Ouyang, R. Jiao, Z. Chen, S. Zhang, J. Yang, L. Mo, J. Zeng, W. Xue *et al.*, "Unsupervised detection of microservice trace anomalies through service-level deep bayesian networks," in *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2020, pp. 48–58.

[49] X. Guo, X. Peng, H. Wang, W. Li, H. Jiang, D. Ding, T. Xie, and L. Su, "Graph-based trace analysis for microservice architecture understanding and problem diagnosis," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020, pp. 1387–1397.

[50] C. Hou, T. Jia, Y. Wu, Y. Li, and J. Han, "Diagnosing performance issues in microservices with heterogeneous data source," in *2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*. IEEE, 2021, pp. 493–500.

[51] L. Zheng, Z. Chen, J. He, and H. Chen, "Mulan: multi-modal causal structure learning and root cause analysis for microservice systems," in *Proceedings of the ACM Web Conference 2024*, 2024, pp. 4107–4116.

[52] C. Zhao, M. Ma, Z. Zhong, S. Zhang, Z. Tan, X. Xiong, L. Yu, J. Feng, Y. Sun, Y. Zhang *et al.*, "Robust multimodal failure detection for microservice systems," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 5639–5649.

[53] Y. Sun, T. Zheng, X. Wen, W. Kuang, H. Liu, S. Zhang, C. Shen, B. Wu, and D. Pei, "A multimodal intelligent change assessment framework for microservice systems based on large language models," 2025.

[54] G. Yu, P. Chen, Y. Li, H. Chen, X. Li, and Z. Zheng, "Nezha: Interpretable fine-grained root causes analysis for microservices on multi-modal observability data," in *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2023, pp. 553–565.