FAMOS: Fault diagnosis for Microservice Systems through Effective Multi-modal Data Fusion

Chiming Duan*, Yong Yang‡, Tong Jia†§, Guiyang Liu¶, Jinbu Liu¶,
Huxing Zhang¶, Qi Zhou¶, Ying Li‡ and Gang Huang§
*School of Software and Microelectronics, Peking University, Beijing, China
duanchiming@stu.pku.edu.cn
†Institute for Artificial Intelligence, Peking University, Beijing, China
jia.tong@pku.edu.cn
‡National Engineering Research Center for Software Engineering, Peking University, Beijing, China
{li.ying, yang.yong}@pku.edu.cn
§National Key Laboratory of Data Space Technology and System, Beijing, China
{jia.tong, hg}@pku.edu.cn
¶Alibaba Cloud Computing Company, Hangzhou, China
{wuming.lgy, liujinbu.ljb, huxing.zhx, jackson.zhouq}@alibaba-inc.com

***Abstract*—Accurately diagnosing the fault that causes the failure is crucial for maintaining the reliability of a microservice system after a failure occurs. Mainstream fault diagnosis approaches are data-driven and mainly rely on three modalities of runtime data: traces, logs, and metrics. Diagnosing faults with multiple modalities of data in microservice systems has been a clear trend in recent years because different types of faults and corresponding failures tend to manifest in data of various modalities. Accurately diagnosing faults by fully leveraging multiple modalities of data is confronted with two challenges: 1) how to minimize information loss when extracting features for data of each modality; 2) how to correctly capture and utilize the relationships among data of different modalities. To address these challenges, we propose FAMOS, a Fault diagnosis Approach for MicrOservice Systems through effective multi-modal data fusion. On the one hand, FAMOS employs independent feature extractors to preserve the intrinsic features for each modality. On the other hand, FAMOS introduces a new Gaussian-attention mechanism to accurately correlate data of different modalities and then captures the inter-modality relationship with a cross-attention mechanism. We evaluated FAMOS on two datasets constructed by injecting comprehensive and abundant faults into an open-source microservice system and a real-world industrial microservice system. Experimental results demonstrate the FAMOS's effectiveness in fault diagnosis, achieving significant improvements in F1 scores compared to state-of-the-art (SOTA) methods, with an increase of 20.33%.**

***Index Terms*—Microservice systems, multi-modal data, fault diagnosis**

I. INTRODUCTION

In recent years, microservice architecture has been widely adopted in industry due to its loose coupling, flexibility and scalability. Microservice architecture decomposes a large and complex system into multiple small services that can be independently developed, deployed and managed. These services cooperate with each other to fulfill the system functions. A typical microservice system involves complicated hardware and software environment and intricate interactions among services. A software or hardware fault from one service can propagate through the microservice topology, ultimately causing failures within the entire system, which has become a significant challenge to guarantee the reliability of microservice systems. Therefore, accurately diagnosing the fault that causes the failure is crucial for Site Reliability Engineers (SREs) to formulate error recovery strategies to maintain the reliability of microservice systems.

Fault diagnosis **[1]–[15]** has been widely studied in recent years. These methods are data-driven and mainly rely on three modalities of runtime data: traces, logs, and metrics. Different types of faults and corresponding failures tend to manifest in data of various modalities [16]. Data of a single modality is not capable of revealing the root causes of failures with complicated software and hardware environment and service dependencies. Therefore, diagnosing faults with multiple modalities of data has obtained much attention in industrial and academia. Numerous attempts [9]–[15] have been made to employ multi-modal data fusion techniques to combine the three modalities of data to achieve better fault diagnose efficacy. Depending on the timing of data fusion, these methods can be categorized into early-fusion-based methods and late-fusion-based methods. Early-fusion-based methods first convert the collected raw data from different modalities into a unified format through feature engineering techniques. Examples include transforming the data into multi-dimensional time series [11] or events [9], [10], [12], after which the unified format data is fed into a single model for fusion. In contrast, late-fusion-based methods independently extract or analyze information from each modality to generate intermediate results, which are then integrated [13] or fused [14], [15] to produce the final output.

Despite all the efforts that have been made, accurately

* Work was done when Chiming was an intern at Alibaba.
* Ying Li is the corresponding author.

1558-1225/25/$31.00 ©2025 IEEE
DOI 10.1109/ICSE55347.2025.00073

diagnosing faults with multiple modalities of data is still confronted with two challenges: 1)how to minimize information loss when extracting features for data of each modality; 2)how to correctly capture and utilize the relationships among data of different modalities. For early-fusion-based methods, the structure differences among the three modalities' data can result in the loss of intrinsic information during fusion process. On the one hand, converting raw data with different structures into a unified format may lead to information loss. For example, converting tree-structured traces into time series results in the loss of their unique topological structure. On the other hand, a single model may struggle to effectively handle and extract comprehensive information from all different modalities' data simultaneously. For instance, Graph Attention Networks [17] (GATs) are adept at processing graph-structured data (trace) but may not readily extract temporal information from time series (metrics). For late-fusion-based methods, they fail to correctly capture and utilize the complex relationships among modalities. First, because data from each modality is processed independently, the inter-modal relationships are not considered or utilized during the processing, resulting in intermediate results that lack inter-modal relationships. Second, existing methods often handle intermediate results through integration [13] or naive fusion techniques such as concatenation [14], which fail to effectively extract inter-modal relationships.

To address the challenges above, we propose FAMOS, a novel fault diagnosis approach for microservice systems with muti-modal data. For the first challenge, FAMOS adopts a late-fusion-based paradigm and employs independent feature extractors for each modality of data, which are tailored to the structures of each modality. In this way, the intrinsic information of each modality is well preserved for later data fusion. To address the second challenge, we introduce a new Gaussian-attention mechanism, which assigns weights based on the degree of correlations between modalities during data processing. This ensures that the intermediate results of the processing incorporate inter-modal data relationships. Moreover, during the fusion phase, we utilize a cross-attention mechanism to extract the inter-modal data relationships by computing cross-modal attention. By constructing datasets through fault injection, we evaluated FAMOS's performance on an open-source microservice system and a real-world industrial microservice system. Experiments on both datasets demonstrate the effectiveness of FAMOS, achieving an average F1 score of 85.69%, which surpasses existing state-of-the-art methods by 20.33%. The contributions of this paper can be summarized as follows:

- We propose FAMOS, a novel fault diagnosis approach that fully leverages multi-modal data, preserving intricate information within each modality and capturing relationships between modalities.
- By employing Gaussian-attention and Cross-attention mechanisms, FAMOS effectively extracts inter-modal relationships during the late fusion process, thereby enhancing the performance of fault diagnosis.
- Through a comprehensive and abundant fault injection, we have constructed two datasets from an open-source microservice system and real-world industrial microservice system. FAMOS is thoroughly evaluated on the datasets and the datasets are made public for research.
- Experiments on the datasets have demonstrated the effectiveness of FAMOS, which surpasses existing state-of-the-art methods by 20.33% in F1 score.

II. PRELIMINARIES

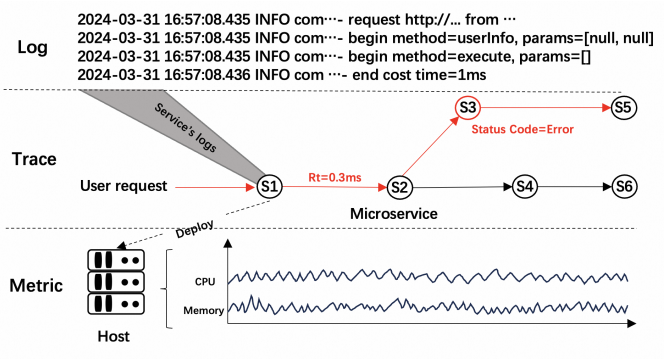


Fig. 1: Multi-modal data in microservices

A. Traces, Metrics, Logs

To ensure the reliability of microservices, site reliability engineers (SREs) typically collect three modalities of runtime data to monitor the system's operational status: traces, logs, and metrics [18].

As illustrated in Figure 1, **traces** capture the service invocation relationships during a user request, forming a tree-like structure. In this context, an edge (S_1, S_2) represents an invocation from service S_1 to service S_2 within that request. Each trace is identified by a unique trace ID, while the individual nodes within the trace are distinguished by separate span IDs. In addition to the invocation relationships, the trace records the response time (RT) between services and the status code indicating whether the invocation was successful. Overall, traces offer a holistic view of how a user's request is processed through the whole miscroservice system. This enhances transparency in system behavior and aids in the maintenance and optimization of the system. **Logs** are semi-structured textual information output by the microservices system. They typically consist of three parts: a timestamp, log level, and content. The timestamp records the time at which the log was generated. Log levels are usually set to indicate the service's operational status at the time of logging. The log content records operational events of the system, containing rich semantic information. The transition between logs records the sequential information of system status. **Metrics** record the performance indicators of the system, such as CPU utilization and memory usage. SREs collect these performance indicators at a specific sampling frequency, encompassing the physical hosts or virtual machines where the microservices

are deployed, and organize them into multi-dimensional time series. Metrics contain rich temporal information, representing the fluctuations in system performance over a certain time range. Traces, metrics, and logs each have distinct structures, encompassing spatial, temporal, and semantic information. In practice, SREs typically combine these three aspects of information to comprehensively monitor and manage the operational state of microservices systems.

B. Fault Diagnosis

In a microservices system, system defects caused by code design issues (such as circular dependencies) or hardware problems (such as insufficient CPU resources) are referred to as faults [19]. When a user request triggers one or more faults, the resulting changes in the service are termed errors. These errors can propagate internally within a service or externally between services. When errors reach the service interface and affect the service provided by the system, they result in a failure. A failure manifests as user request errors or timeouts. When a failure occurs, SREs need to promptly diagnose the faults that caused the failure to implement the necessary recovery measures. Fault diagnosis aims to infer the concrete type of the fault and map it to a predefined set of fault types, such as CPU overload. Essentially, our task is a multi-class problem. Specifically, given three modalities of data: traces $\mathbf{T}$, metrics $\mathbf{M}$, and logs $\mathbf{L}$, as well as a predefined set of fault types $\mathbf{F}$, our task is to find a mapping relation: $f : (\mathbf{T}, \mathbf{M}, \mathbf{L}) \rightarrow \mathbf{F}$. By providing specific fault types, fault diagnosis aids SREs in making quicker decisions and formulating effective recovery strategies, which is crucial for maintaining the reliability of microservices systems.

III. METHOD

A. Overview

In this paper, we propose FAMOS, a novel fault diagnosis method that fully leverages multi-modal data, avoiding information loss and lack of inter-modal relationship extraction. Figure 2 shows the overview of FAMOS. FAMOS comprises three main stages: data alignment, feature extraction, and data fusion. In order to avoid information loss, our method employs late fusion, where we use the most suitable feature extractors for each modality to extract information. Specifically, during the data alignment stage (Sec. III-B), our goal is to correlate different multi-modal data to avoid the loss of inter-modal relationships that might occur during independent processing of each modality's data. Since traces record the complete path of a user's request within a microservice, they provide a fine-grained depiction of the microservice system's behavior during a single invocation. Therefore, we employ a request-based alignment method, correlating logs and metrics with traces. This correlation is then incorporated as additional features into the independent processing of each modality. For logs, we distinguish logs from different invocation processes and organize them into multiple log sequences. We extract information from each log sequence to ensure that the generated intermediate results correspond to specific traces. For metrics, we designed a Gaussian attention mechanism to facilitate alignment, assigning higher weights to metric segments that have a stronger correlation with the trace. By employing a request-based alignment strategy, we correlate different modalities to specific system operations and events. This not only precisely establishes the relationships between modalities but also helps capture contextual information, leading to a better understanding and analysis of system behavior. In the feature extraction stage (Sec. III-C), we extract information from each modality independently. We design different feature extractors for the three modal data based on their structural characteristics, thoroughly extracting the intrinsic information in each data modality. In the data fusion stage (Sec. III-D), we use a cross-attention mechanism to compute the attention between different modalities. This allows us to mine the complex relationships between multi-modal data, thereby uncovering hidden patterns across different modalities and improving the model's effectiveness. In summary, FAMOS leverages the advantages of multi-modal data effectively, preventing information loss and addressing the challenge of extracting inter-modal relationships, thereby significantly enhancing the efficiency of fault diagnosis task.

B. Data Alignment

During the data alignment stage, our task is to associate different modalities of data to ensure that these data can collectively reflect the system's state at the same time point or within the same event context. Previous methods [9], [14], [16], [20] adopted a time-based alignment approach, correlating data within the same time window. Specifically, these methods relied on the entire microservice topology to aggregate multi-modal data information within a certain time window, including the invocation details of a particular service, the metrics of hosts and nodes where the service is deployed, and the logs generated by the service within that period. However, this approach has some limitations and cannot accurately correlate different modalities data. For instance, it may cause fragmentation at the edges of time windows, failing to capture complete traces. Additionally, since multiple user requests may be concurrent at the same time, a time-based alignment approach cannot accurately describe the specific request paths, and the alignment process may introduce some redundant information. For example, in a Kubernetes-based microservice environment, a microservice S can be deployed across multiple Pods, such as Pod A and Pod B. Each request may be handled by different Pods. If requests within a certain time window only occur on Pod A, then the metrics from Pod B during this period is redundant.

Unlike previous work, FAMOS adopts a request-based alignment approach. Since each trace corresponds to a single user request, we achieve the association of multi-modal data with a user request by aligning metrics and logs to traces. By aligning multi-modal data to trace, we can clearly view all events and performance metrics occurring during the complete service invocation, thereby facilitating a more accurate fault diagnosis. Specifically, we inject the current thread's trace ID

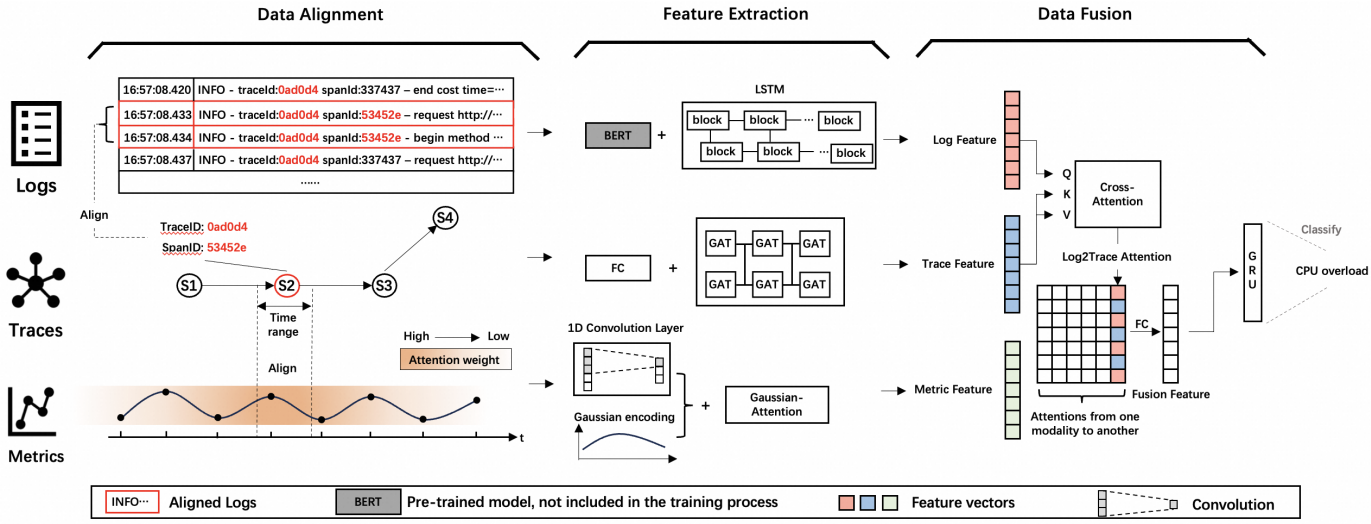


Fig. 2: The overview of FAMOS.

and span ID into the logging framework, ensuring that the log output format includes these two IDs. This allows us to associate logs with specific traces. By using these IDs in the logs, we can organize logs dispersed across different microservices into logically ordered sequences. Each log sequence is associated with a particular trace. When we extract each log sequence independently, the intermediate results still maintain their association with the specific trace, thereby preserving relationships. For metrics, we associate the metrics with the trace through the host IP and pod IP according to the hosts and virtual machines where the service is deployed. Finally, we completed the request-based alignment. Each aligned sample can be represented as a graph $\mathcal{G} = \{V, E\}$. Here, $V = \{v_1, v_2, \ldots, v_l\}$ denotes the service interface nodes traversed by the request, and $E = \{(v_a, v_b), \ldots, \}$ represents the invocation path between these nodes, effectively illustrating the service request path. Each node $v_i = \{\mathbf{T}_i, \mathbf{L}_i, \mathbf{M}_i\}, v_i \in V$, comprises data from three modalities: traces ($\mathbf{T}_i$), logs ($\mathbf{L}_i$), and metrics ($\mathbf{M}_i$). Next, we will extract information from the three modalities based on their respective characteristics and map them into the same feature space. This facilitates subsequent modality fusion and downstream tasks.

C. Feature Extraction

During the feature extraction stage, our objective is to map the aligned multi-modal data with different structures into the same feature space, ensuring they have similar representations while simultaneously preserving intrinsic information in each modal. Among the three modalities data, traces contain the service dependency relationships and the invocation details of each service, representing spatial information. Logs are organized textual information generated by service operations, containing rich semantic information. Metrics include performance indicators such as CPU usage from nodes where services are deployed. These metrics form a multi-dimensional time series, encapsulating rich temporal information. Previous methods did not fully consider the different structures of modal data when extracting features, leading to information loss during the extraction process, such as the loss of semantic information [14]. In contrast, We design the most suitable feature extractors tailored to the structures of different modal, thereby avoiding information loss.

1) Trace Extraction: For traces, we use GAT [17] to extract its spatial information. GAT dynamically assigns weights to neighbors by learning the representations of nodes and edges, effectively capturing global topological relationships. For an aligned sample $\mathcal{G} = \{V, E\}$, the input to GAT consists of trace invocation path E and trace-related information of each node. The trace-related information contains three features: the request time for the service interface, the request status code, and the service interface name. Since we construct dependency relationships based on specific request paths rather than the entire microservice topology, we also include the service interface name as a node feature to uniquely identify each node. In order to integrate the three features, we use BERT [21] to encode the service interface names, and then pass the three features through a fully connected layer to obtain a unified representation. It is worth noting that, due to the limited number of service interface names in microservices, we store the mappings from service interface names to word embeddings and retrieve the mappings for repeated instances. Finally, we feed this integrated representation into the GAT network to learn the global representation. Given the integrated trace feature $\mathbf{T}_i$ of node i, the GAT-encoded feature $\mathcal{T}_i$ can be obtained as follows:

$$\mathcal{T}_i = \text{ReLU}\left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij} \mathbf{W} \mathbf{T}_i\right), \tag{1}$$

where $\mathbf{W}$ represents the learnable parameters, $\mathcal{N}(i)$ denotes the nodes connected to node i, and α_{ij} is the attention weight

for the edge (v_i, v_j), which can be obtained by the following equation:

$$\alpha_{ij} = \frac{\exp(\text{LeakyReLU}\left(\mathbf{a}^T\left[\mathbf{W}\mathbf{T}_i \| \mathbf{W}\mathbf{T}_j\right]\right))}{\sum_{k\in\mathcal{N}(i)}\exp(\text{LeakyReLU}\left(\mathbf{a}^t\left[\mathbf{W}\mathbf{T}_i \| \mathbf{W}\mathbf{T}_k\right]\right))}, \quad (2)$$

where $\mathbf{a}^T$ is the learnable attention vector, and $\|$ denotes the vector concatenation operation. By employing GAT encoding, the trace modal data of nodes is transformed into a vector that encapsulates the comprehensive information of the nodes. This vector is capable of reflecting the nodes' positions and relative relationships within the entire graph.

2) Log Extraction: For logs, each node's log information consists of multiple log texts. Each log text contains semantic information related to the current invocation. We use BERT [21] to extract the semantic information from the log texts and convert them into word vectors. In addition to semantic information, the order of the log texts is also very important [22], [23]. Therefore, we further use LSTM to encode the embeddings of the logs to capture the sequential relationship of their occurrences. Let the log embeddings of node i be $\mathbf{L}_i$, the output $\mathcal{L}_i$ after LSTM encoding can be expressed as:

$$\mathcal{L}_i = \text{LSTM}(\mathbf{L}_i), \quad (3)$$

3) Metric Extraction: For metrics, we first follow previous works [7], [14] by using a one-dimensional convolutional network to integrate different dimensions of the metrics. This approach allows for an accurate modeling of multivariate time series, providing a reasonable representation. In order to ensure that the association relationship is preserved in the intermediate results, we will use Gaussian-attention for further extraction, the process will be shown in section III-D1.

D. Data Fusion

During the data fusion phase, our objective is to fuse data from different modalities for downstream tasks. Our objectives are twofold: first, to accurately associate data from different modalities, and second, to extract the relationships between these modalities. To achieve these goals, we employ two distinct attention mechanisms, each tailored to address one of these tasks.

1) Gaussian-attention: We use the Gaussian-attention mechanism to establish modal relationships between metric and trace. Specifically, we use a Gaussian distribution to assign weights when calculating the attention to various metric segments. This ensure that metric segments more relevant to the current user request (i.e., those happening closer in time) receive higher weights. For a trace passing through microservice S_1, we associate the relevant metrics of S_1 with that trace. Given a segment of multi-dimensional time series metrics $M = \{m_1, m_2, \ldots, m_k, \ldots, m_t\}$, where sampling point m_k lies within the execution time range of the trace and is the most relevant sampling point for the trace. The Gaussian weight $G(m_i)$ for a sampling point m_i is defined as:

$$G(m_i) = f(i; k, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{|i-k|^2}{2\sigma^2}\right), m_i \in M, \quad (4)$$

In Equation 4, the sampling points closer to the execution time range of the trace have higher Gaussian weights. During alignment, we incorporate the Gaussian weights as an additional feature into the metrics. In the feature extraction stage (Sec. III-C3), we use the Gaussian weights for the positional encoding of the metrics and extract their features through attention mechanisms. This allows us to give higher weights to segments more closely relevant to the trace during extraction, thereby preserving the inter-modal relationships in the intermediate results. Next, we employ a Gaussian-attention mechanism to extract the temporal information from the metrics, mapping it to a feature space consistent with other modalities. Inspired by positional encoding in Transformers [24], we incorporate Gaussian weights as a form of positional encoding into the attention computation. In this way, our model focuses more on the parts with higher Gaussian weights during feature extraction, which correspond to segments closer to the trace execution times. Specifically, given the metric features $\mathbf{M}_i$ of node i extracted by the one-dimensional convolutional network, the output of the attention encoding $\mathcal{M}_i$ can be obtained by the following formula:

$$\mathcal{M}_i = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}, \quad (5)$$

where the query matrix $\mathbf{Q}$, key matrix $\mathbf{K}$, and value matrix $\mathbf{V}$ are obtained by the following equations:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_Q, \mathbf{K} = \mathbf{X}\mathbf{W}_K, \mathbf{V} = \mathbf{X}\mathbf{W}_V, \mathbf{X} = \mathbf{M}_i + G(\mathbf{M}_i), \quad (6)$$

Here, $\mathbf{W}_Q$, $\mathbf{W}_K$, and $\mathbf{W}_V$ are trainable parameters, and $G(\mathbf{M}_i)$ represents the Gaussian weights of $\mathbf{M}_i$. By employing Gaussian encoding to precisely highlight the segments of metrics that are most relevant to traces, we can extract the temporal information of metrics into feature vectors while preserving their relationships with traces.

2) Cross-attention: Previous methods have merely combined data from various modalities, lacking the mining of complex inter-modal relationships. Beyond incorporating information from the modal data itself, we also fuse the associative relationships between cross-modal data to capture more comprehensive information representations. Specifically, we employ a cross-attention mechanism to extract the relationships between different modalities. Cross-attention [24] can dynamically adjust the weights of each modality, effectively filtering out irrelevant information and highlighting relevant information. This makes it especially suitable for extracting relationships between different modalities. Specifically, for cross-modal attention from log to trace, the input is the log features $\mathcal{L}_i$ and the trace features $\mathcal{T}_i$, and the resulting attention $Attn(\mathcal{L}_i, \mathcal{T}_i)$ is given by the following equation:

$$\text{Attn}(\mathcal{L}_i, \mathcal{T}_i) = \text{softmax}\left(\frac{(\mathcal{L}_i \mathbf{W}'_Q) \cdot (\mathcal{T}_i \mathbf{W}'_K)^T}{\sqrt{d_k}}\right)(\mathcal{T}_i \mathbf{W}'_V), \tag{7}$$

Here, $\mathbf{W}'_Q$, $\mathbf{W}'_K$, and $\mathbf{W}'_V$ are learnable parameters. Thus, we can compute the cross-modal attention for the three different modalities. Finally, we use a fully connected layer to interleave the different attentions, obtaining the fused feature X_i for node i. Given that the number of nodes included in each call can vary, we employ a Gated Recurrent Unit (GRU) as a classifier to adapt to the dynamically changing nodes and classify the fused features of different nodes:

$$\mathbf{f} = \text{GRU}(X_1, X_2, ..., X_n), f \in \mathbf{F}, \tag{8}$$

Where $\mathbf{f}$ denotes the failure type mapped to the predefined category $\mathbf{F}$, and n represents the number of nodes. Finally, we map the results to our predefined fault types, completing the fault diagnosis task.

IV. EXPERIMENTS

We conduct our experiments focuing on three following research questions (RQS):

- **RQ1:** How does FAMOS performs in terms of effectiveness?
- **RQ2:** Does data from each modality contribute to FAMOS?
- **RQ3:** Does each main component contribute to FAMOS?
- **RQ4:** How sensitive is the Gaussian-attention component of FAMOS to variations in its variance?

A. Dataset

Existing open-source microservice fault datasets have certain limitations:

- **Limited data sources and insufficient fault types**: Some datasets [25] only include uni-modal data. Other datasets [14] are designed for localization tasks, pinpointing the services where faults occur, and have a limited variety of fault types (with only three types of faults).
- **Lack of real-world industrial datasets**: Most datasets are based on open-source benchmark microservice systems, making it difficult to reflect real industrial environments.

To explore the performance of FAMOS in fault diagnosis, particularly in real-world industrial scenarios, we constructed two datasets through fault injection on both a benchmark microservice system and a real industrial system. Our objective is to construct two multimodal datasets with a rich variety of fault types, making it suitable for our tasks.

1) Microservice system and Data collection: We collect our data on an open-source microservice train-ticket [26] and an industrial microservice referred to as Mall. Train-ticket is a benchmark microservice system providing railway ticket services, where users can perform operations such as ticket booking, payment, and rescheduling. Mall offers various e-commerce functionalities, such as viewing product details and obtaining customer reviews about products. Both microservices are deployed in a Kubernetes [27] environment.

For the train-ticket microservice, we used Locust [28] to simulate user requests, covering functionalities like route selection and ticket booking. We set the concurrent user request count to 10 with a startup speed of 1 request per second. For Microservice A, to minimize disruption to overall business operations, we conducted fault injection and collected data during a gray release phase, incorporating real-world user request data.

We employed our observability platform, referred to as P, to collect data from three modalities. Compared to some open-source collection tools, P gathers data more precisely. Specifically, it has the following features: At the trace level, we incorporate method stacks that can help unravel some faults into the trace request path. These method stacks include various database call methods such as GET and POST, which can more accurately reveal root causes when faults occur. At the log level, we injected the trace ID and span ID of the current thread into the logging framework, achieving precise association between logs and traces. For metrics, we collected data based on Prometheus [29]. By using the query language PromptQL, we extracted and aggregated data from Prometheus's time series database, then locally collected the metrics. We primarily gathered two levels of information: host metrics from the deployment environment of the microservice, and container metrics specific to the microservice itself, encompassing 41 types of metrics such as CPU usage and memory usage. By correlating the pod IP and host IP information from the trace data, we achieved the association of metrics at both levels with the current request. Our data can be found at: https://modelscope.cn/datasets/dcm552/FAMOS-dataset.

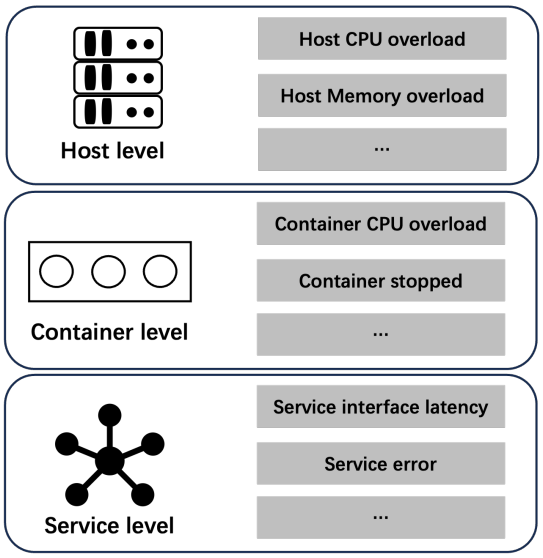


Fig. 3: Faults at Different Levels

2) Fault injecting and Labeling: We used ChaosBlade [30] to inject faults into Train-Ticket and Microservice A. As shown in figure 3, we performed fault injection at three levels:

host-level, container-level, and service-level, simulating faults that could occur at these different levels. In total, we injected 16 types of faults into the two microservice systems, with each fault injection session lasting from 5 to 10 minutes, and the entire fault injection process spanning 6 days. Subsequently, we obtained three modalities of data from the microservices along with their corresponding data labels. These labels include the time periods when faults were injected and the locations where the faults occurred.

B. Competitors

For RQ1, in order to investigate the effectiveness of the FAMOS method, we compare it with state-of-the-art (SOTA) fault diagnosis methods, including those utilizing unimodal data [4], [6] and those based on multimodal data [9], [15]. Note that our comparison of the proposed unimodal method is intended to demonstrate the necessity of multimodality. In contrast, the comparison with multimodal methods aims to validate the effectiveness of the FAMOS itself.

- **DiagFusion** [9] is a early-fusion-based methods. It employs traces, logs, and metrics for fault diagnosis. It converts data from different modalities into unified events and uses embedding techniques to obtain consistent representations.
- **CloudRCA** [15] is a late-fusion-based methods. It leverages topology, logs, and metrics information to detect faults in cloud computing platforms. After integrating this information, CloudRCA utilizes the KHBN network to identify system faults.
- **CloudRanger** [6] combines topology and metrics information to perform causal inference through the correlation between services, thereby identifying faults in IBM's commercial cloud.
- **CloudLog** [4] bases its fault diagnosis on ERROR-level logs and topology information, identifying critical paths in the cloud operating system to classify faults.

For RQ2, we examined the significance of each modality of data in fault diagnosis by individually ablating the Trace-related information (without trace info), Log (without log), and Metric (without metric) modalities. Of note, due to our approach of associating logs and metrics with traces, we are unable to completely ablate the topological information of traces. Instead, we can only ablate trace-related information.

For RQ3, to explore the importance of different components of FAMOS, we conducted comparisons with various variants. Specifically, we ablated Gaussian-attention and Cross-attention, resulting in the variants FAMOS (w/o Gaussian-attention) and FAMOS (w/o Cross-attention), in order to investigate the contribution of these two attention mechanisms in the alignment and fusion stages. Additionally, we examined the significance of each modality's feature extractor. We conducted ablation studies by separately removing the log semantic extractor (w/o BERT), the metrics temporal extractor (w/o 1D Convolution), and the trace topology extractor (w/o GAT) to investigate whether our designed feature extractors effectively preserve the information of each modality.

For RQ4, in order to explore the sensitivity of the Gaussian-attention mechanism to the variance σ in Equation 4, we compared the performance of FAMOS under different values of σ. Specifically, we selected Gaussian variance values ranging from 1 to 10, with an increment of 1. This exploration allows us to understand the robustness of the Gaussian-attention mechanism within FAMOS.

C. Experiment Settings and Implementation Details

We partitioned the dataset into training and test sets based on a time-based split with a ratio of 7:3. Within the training set, we further randomized the division into training and validation subsets with a ratio of 8:2. By splitting the training and test sets based on time, we avoid the risk of data leakage. Moreover, the random division of the validation subset ensured the robustness of the model obtained through hyperparameter tuning.

Our experiments were conducted on a Linux server equipped with an Intel(R) Xeon(R) Platinum 8369B CPU @ 2.90GHz, an NVIDIA A10 GPU with 24GB of GPU memory, and 64GB of system memory. The hyperparameters for our experiments are listed in Table I. We used the Adam optimizer [31], with the number of training epochs set to 50 and the learning rate set to $1e^{-4}$. All structures leveraging multi-head attention mechanisms (GAT, Gaussian-attention, Cross-attention) employed 4 attention heads. Except for the LSTM and fully connected layers, all network architectures we employed have one layer to enhance model speed. For RQ1, RQ2 and RQ3, we set the variance σ in the Gaussian-attention mechanism to 5.

During the feature extraction phase, our word embedding dimension was set to 1024, using a two-layer LSTM structure with a hidden size of 128. When extracting trace features, we used a three-layer fully connected (FC) network with a hidden dimension of 128. The convolutional layer had 64 channels, and the kernel size was set to 2. For the fusion stage, the output feature dimension was set to 64. Our code can be accessed via https://github.com/alibabacloud-observability/FAMOS.

TABLE I: Hyperparameters

Hyperparameter	Value
epoch	50
batch size	512
learning rate	$1e^{-4}$
Attention heads	4
Word embedding size	1024
LSTM layer	2
LSTM hidden size	128
FC layer	3
FC hidden size	128
Feature extraction output size	64
Fusion feature size	64
σ in Gaussian-attention	5.0

D. Measurements

To comprehensively evaluate the effectiveness of FAMOS, we employ three key performance metrics: Macro Precision, Macro Recall, and Macro F1-score. These metrics provide a balanced assessment across all classes by equally weighted each class's performance, which is crucial when dealing with imbalanced datasets.

Macro Precision quantifies the accuracy of our system's predictions for each class and then averages these precisions across all classes. It is calculated as the arithmetic mean of the precision values for each class, mathematically represented as:

$$\text{Macro Precision} = \frac{1}{C}\sum_{i=1}^{C}\frac{TP_i}{TP_i + FP_i}, \tag{9}$$

where C is the total number of fault classes, and TP_i and FP_i are the true positives and false positives for class i, respectively.

Macro Recall measures the system's ability to identify all instances correctly across all classes by averaging the recall values for each class. It is computed as:

$$\text{Macro Recall} = \frac{1}{C}\sum_{i=1}^{C}\frac{TP_i}{TP_i + FN_i}, \tag{10}$$

where TP_i and FN_i are the true positives and false negatives for class i, respectively.

The Macro F1-score offers a balanced view by calculating the harmonic mean of Macro Precision and Macro Recall, effectively combining both metrics into a single metric that accounts for both false positives and false negatives. The formula is:

$$\text{Macro F1} = \frac{1}{C}\sum_{i=1}^{C}\frac{2\cdot(\frac{TP_i}{TP_i+FP_i}\cdot\frac{TP_i}{TP_i+FN_i})}{\frac{TP_i}{TP_i+FP_i}+\frac{TP_i}{TP_i+FN_i}}, \tag{11}$$

E. Evaluation Results

1) ***RQ1:*** *How does FAMOS performs in terms of effectiveness?:* The comparison between FAMOS and the baselines is shown in Table II. In the table, we use boldface to highlight the highest metric values and underline to denote the second-highest metric values. Across all three performance metrics, FAMOS outperforms existing methods, demonstrating its high effectiveness. On one hand, FAMOS uses Gaussian-attention to correlate data from different modalities, thereby providing a fine-grained characterization of the specific behaviors during the user request process in a microservice system, which better highlights system characteristics when a fault occurs. On the other hand, FAMOS fuses multi-modal data using Cross-attention, capturing not only the information within each modality but also the relationships between different modalities, thus achieving efficient fault diagnosis.

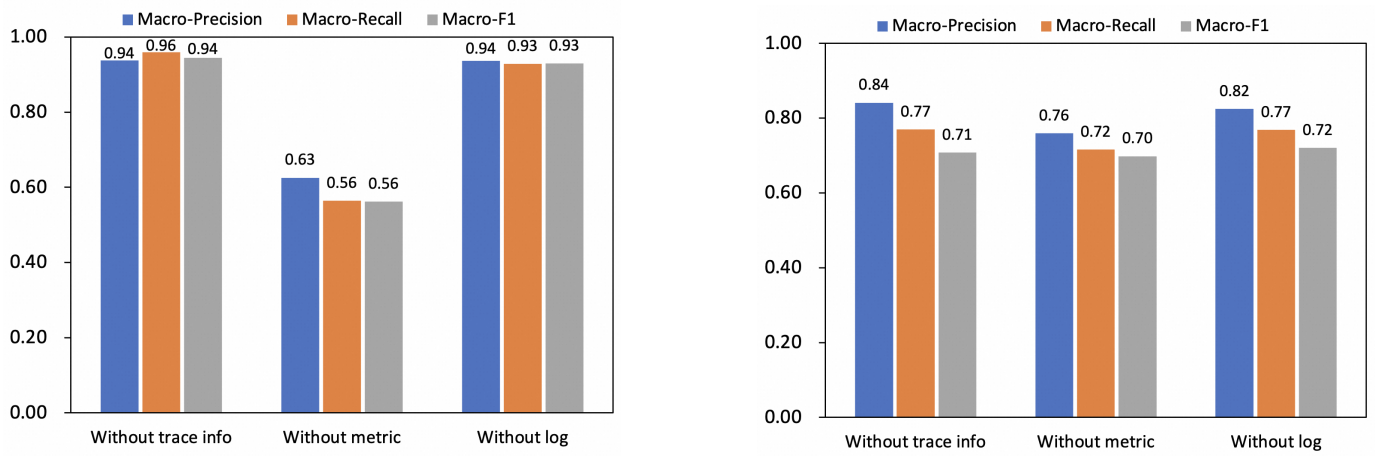
Among the baseline methods, DiagFusion [9] achieved the best results, attaining an average F1 score of 65.37% on two datasets. DiagFusion addresses the problem of unified representation of multi-modal data by representing different modalities as events in the same semantic space. However, extracting representations of different multi-modal data into unified events can result in information loss. Specifically, for the metrics modality, DiagFusion monitors anomalies using the 3-sigma method and extracts them as either increasing or decreasing events. This extracting approach, however, loses information about the degree and speed of metric fluctuations. Due to the system's fault tolerance and recovery mechanisms, fluctuations within a certain range might not lead to failures. In contrast, FAMOS designs corresponding feature extractors based on the structure of each modality. For metrics, FAMOS utilizes one-dimensional convolutional networks and attention mechanisms to accurately extract multidimensional temporal information, thereby making more precise judgments about the system's state. Regarding the F1 score, FAMOS surpassed DiagFusion [9] by an average of 20.33%.

CloudRCA is another method that utilizes multimodal data to infer faults. It first detects anomalies in the metrics and logs of various services and constructs a Bayesian network using the topological relationships between services. Then, CloudRCA employs a Knowledge-informed Hierarchical Bayesian Network (KHBN) to infer system faults by integrating the anomalous information from each modality. However, due to the challenges Bayesian networks face in handling missing data or latent variables [32], they are often unable to accurately capture the relationships between multimodal data. Consequently, CloudRCA performed poorly on two datasets, with an average F1 score of only 57.86%. Unlike CloudRCA, FAMOS fuses multimodal data through a Cross-attention mechanism, accurately capturing the dependencies between different modalities, thereby enabling better inference of system faults.

CloudRanger [6] and CloudLog [4] are two methods that use unimodal data for fault diagnosis. CloudRanger relies on metric data and employs a dynamic causal relationship analysis approach to infer faults without requiring predefined system topology. On the other hand, CloudLog uses system error-level logs and topology for fault diagnosis. However, different types of faults often manifest in different modalities of data, making it difficult for methods that rely on a single modality to infer all faults [16]. As a result, they perform poorly on two datasets, with average F1 scores of only 41.47% for CloudRanger and 20.53% for CloudLog. It is noteworthy that CloudRanger outperforms CloudLog in terms of F1 score, particularly on the Train-ticket dataset, where CloudRanger's F1 score is 34.82% higher than that of CloudLog. This finding suggests that, on one hand, many of the injected faults, such as CPU overload and memory overload, can manifest through metrics. On the other hand, the logs we collected are mostly application-level logs, meaning they originate from within the services themselves. These logs rarely relate to the underlying components (such as the host) and primarily reflect service-level faults. Additionally, we found that the number of logs in the Train-ticket dataset is very sparse, with some services having no relevant log outputs at all, making it more challenging for CloudLog to achieve high performance.

TABLE II: Overall result on Train-ticket and Microservice A datasets

Method	Data Source	Train-ticket			Mall		
		Macro-Precision	Macro-Recall	Macro-F1	Macro-Precision	Macro-Recall	Macro-F1
Ours	multi-modal	**96.39%**	**95.38%**	**95.65%**	**87.41%**	**78.77%**	**75.73%**
DiagFusion	multi-modal	78.48%	76.64%	72.15%	64.89%	64.12%	58.58%
CloudRCA	multi-modal	75.89%	73.46%	68.78%	46.44%	52.41%	46.94%
CloudRanger	metric	63.93%	59.45%	59.73%	23.32%	32.15%	23.20%
CloudLog	log	24.85%	33.47%	24.91%	15.82%	23.17%	16.14%



(a) Train-ticket dataset

(b) Mall dataset

Fig. 4: Modality Data Ablation

This also indicates that there are still differences between existing open-source microservice systems and real enterprise systems. The impact of different modalities of data on our fault diagnosis task will be discussed in detail in the section IV-E2.

2) ***RQ2:*** *Does data from each modality contribute to FAMOS?:* The results of our ablation studies for each data modality are shown in Figure 4. In the ablation of different modalities of data, FAMOS's performance showed varying degrees of degradation. Ablation of the metrics data had the most significant impact, causing an average decrease of 22.68% in F1 scores across the two datasets. Metrics data are crucial in fault diagnosis tasks as they reflect faults at both container and host levels through different metrics. Additionally, faults related to CPU, memory, and other resources are more easily detected in metrics data. However, not all types of faults manifest in metrics data. Some service-level faults are often reflected in the response time of traces and the content of logs. Consequently, ablating trace info and log data led to decreases in F1 scores by 2.99% and 3.19%, respectively. Each of the three modalities of data—metrics, traces, and logs—plays a unique and crucial role in accurately classifying faults. Metrics data provide insight into resource-related issues, while trace data capture the interaction and timing information between services, and log data offer detailed contextual information about service-level events and errors. The combination of these data types enables a more comprehensive understanding of the system's state, leading to more accurate fault diagnosis.

3) ***RQ3:*** *Does each main component contribute to FAMOS?:* Our ablation experiment results are shown in Table III. After individually ablating various components of the model, the performance of FAMOS showed varying degrees of degradation, with an average decrease of 8.94% in F1 score across two datasets. From the ablation results of the model components, removing the Gaussian-attention and Cross-attention mechanisms resulted in a reduction in the F1 score by 5.46% and 2.81%, respectively. The Gaussian-attention mechanism assigns different weights to parts of the metrics, allowing for more accurate alignment with traces. When we replaced Gaussian-attention with Self-attention, it introduced context information irrelevant to the current user request during metrics extraction, thereby affecting the fault diagnosis results. In the data fusion stage, we utilized Cross-attention to accurately capture the dependencies between different modalities by computing cross-modal attention, leading to more comprehensive fault diagnosis. Furthermore, after ablating the feature extractors for each modality, the model experienced an average decrease of 12.15% in F1 score across both datasets. The feature extractors for each modality are designed according to their specific structures to maximize the retention of information from each modality. For instance, when we ablate BERT, the log semantic information is lost,

TABLE III: Ablation Study

Method	Train-ticket			Mall		
	Macro-Precision	Macro-Recall	Macro-F1	Macro-Precision	Macro-Recall	Macro-F1
FAMOS	**96.39%**	**95.38%**	**95.65%**	**87.41%**	**78.77%**	**75.73%**
FAMOS (w/o Gaussian-attention)	90.47%	89.61%	89.39%	80.58%	77.19%	71.07%
FAMOS (w/o Cross-attention)	95.78%	93.58%	94.51%	84.40%	75.95%	71.26%
FAMOS (w/o BERT)	90.66%	82.84%	82.95%	80.46%	73.91%	69.06%
FAMOS (w/o 1D Convolution)	62.84%	60.32%	59.25%	75.30%	70.21%	68.81%
FAMOS (w/o GAT)	92.98%	91.41%	90.86%	84.26%	75.61%	70.34%

resulting in an average decrease of 9.69% in F1 score across the two datasets.

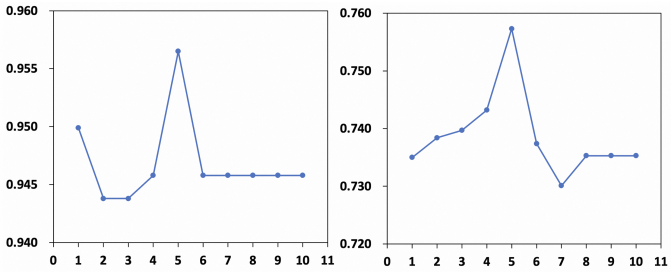


Fig. 5: Different σ values on the Train-ticket dataset (left) and Mall dataset (right)

4) ***RQ4:*** *How sensitive is the Gaussian-attention component of FAMOS to variations in its variance?:* We explored the performance of FAMOS on two datasets under different values of σ. As shown in Figure 5, FAMOS exhibited relatively stable performance across various σ values. Specifically, on both datasets, the performance of FAMOS varied by no more than 2% in terms of the F1 score for different σ values. In FAMOS, selecting different σ values corresponds to using different Gaussian distributions for positional encoding, which affects our ability to accurately extract metric information and reliably associate it with traces. As σ increases, the Gaussian distribution transitions from steep to flat. In other words, as σ increases, the portion of the metrics related to the trace becomes longer. While this provides more contextual information, it also increases the risk of introducing redundant information. In our experiments, with the increase of σ, the F1 score of FAMOS initially increased and then decreased. When σ reached 5, FAMOS achieved the best performance, indicating that the Gaussian weighting at this point had the optimal effect. This suggests that the approach successfully introduced the longest contextual relationships while avoiding redundancy, thereby more comprehensively modeling the relationship between the trace and the metrics.

F. Threats to Validity

Regarding the dataset, our dataset was obtained through fault injection, which may result in fault manifestations that are overly pronounced, potentially failing to fully replicate real-world fault scenarios. Actual faults occur infrequently due to system stability, and sourcing fault-related data from large datasets incurs significant manual labeling costs. Therefore, we opted for fault injection to simulate these scenarios. Additionally, although we employed a real microservice system, it still may not encompass all possible real-world scenarios.

Regarding the method, our approach requires that the logs include traceId and spanId for alignment, which increases the data requirements. As tools like OpenTelemetry [33] gain popularity, we expect the costs associated with injecting these IDs to decrease, thereby making this practice a key trend in the collection of microservice observability data.

Regarding the experiments, while we explored the sensitivity of the Gaussian-attention component to different variance values, further analysis of other hyperparameters is needed to fully understand their impact on model performance. Therefore, future research should consider these aspects for a more comprehensive evaluation.

V. Relate work

Fault diagnosis is crucial for maintaining the reliability of microservices and has been an active research area in recent years [1]–[15], [34]–[37]. Some methods focus on the localization task [1]–[3], [8], [10]–[14], aiming to pinpoint the microservice where the fault occurs. Other methods are dedicated to inferring the specific fault [4]–[6], [34], mapping the fault to a set of predefined fault types. However, these methods fail to fully leverage data from three modalities, resulting in poor effectiveness. Some approaches use only one modality [1]–[4], [6] or two modalities [8] of data. Since different types of faults often manifest in different data modalities, these methods cannot diagnose all fault types. Although some methods [9]–[15] use data from three modalities, they suffer from losses in intrinsic modal information and a lack of inter-modal relationship extraction, thus failing to fully capitalize on multimodal data. Specifically, [9]–[12] utilize early fusion approaches, converting different multimodal data into a unified format, followed by using one modality for data fusion to perform downstream tasks. Diagfusion [9], Nezha [10], and Groot [12] convert data from three modalities into unified events, such as using a 3-sigma approach to identify metric fluctuations and convert them into increasing or decreasing events. TrinityRCL [11] transforms data from

three modalities into time series, for example by aggregating logs based on timestamps and converting them into sequences of log types. However, these conversion processes result in information loss. Converting metrics results in the loss of fluctuation information, and converting logs into aggregated sequences loses inherent semantic information. PDiagnose [13], CloudRCA [15] and Eadro [14] adopt a late fusion approach, independently processing each modality's data but ignoring the extraction of inter-modal relationships.

Unlike previous methods, FAMOS fully leverages multi-modal data. On one hand, FAMOS sets most suitable feature extractors for each modality's structure to ensure the completeness of intrinsic information in each modal. On the other hand, FAMOS accurately models and extracts inter-modal relationships through Gaussian-attention and Cross-attention mechanisms, thereby facilitating a more comprehensive fault diagnosis task.

VI. Conclusion

In this paper, we propose FAMOS, a fault diagnosis approach for microservices based on multi-modal data. To comprehensively infer faults in microservices, it is essential to leverage all three modalities of data. However, existing methods fail to address two key challenges in the multi-modal fault diagnosis task, resulting in poor performance. To fully exploit the multi-modal data, FAMOS employs a late fusion strategy to separately extract the spatial information from traces, the temporal information from metrics, and the semantic information from logs, ensuring that no information is lost within each modality. Additionally, FAMOS utilizes Gaussian-attention and Cross-attention mechanisms to accurately capture inter-modal relationships during data processing and fusion. Our method achieved an average F1 score improvement of 20.33% on both a open-source benchmark microservice and a industrial microservice, demonstrating its effectiveness.

Acknowledgment

This research is supported by Alibaba Group through Alibaba Innovative Research Program.

References

[1] Y. Gan, M. Liang, S. Dev, D. Lo, and C. Delimitrou, "Sage: practical and scalable ml-driven performance debugging in microservices," in *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 135–151. [Online]. Available: https://doi.org/10.1145/3445814.3446700

[2] Y. Gan, G. Liu, X. Zhang, Q. Zhou, J. Wu, and J. Jiang, "Sleuth: A trace-based root cause analysis system for large-scale microservices with graph neural networks," in *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 4*, ser. ASPLOS '23. New York, NY, USA: Association for Computing Machinery, 2024, p. 324–337. [Online]. Available: https://doi.org/10.1145/3623278.3624758

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

[4] Y. Yuan, W. Shi, B. Liang, and B. Qin, "An approach to cloud execution failure diagnosis based on exception logs in openstack," in *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*, 2019, pp. 124–131.

[5] Q. Lin, H. Zhang, J.-G. Lou, Y. Zhang, and X. Chen, "Log clustering based problem identification for online service systems," in *Proceedings of the 38th International Conference on Software Engineering Companion*, ser. ICSE '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 102–111. [Online]. Available: https://doi.org/10.1145/2889160.2889232

[6] P. Wang, J. Xu, M. Ma, W. Lin, D. Pan, Y. Wang, and P. Chen, "Cloudranger: Root cause identification for cloud native systems," in *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, 2018, pp. 492–502.

[7] Z. Li, N. Zhao, M. Li, X. Lu, L. Wang, D. Chang, X. Nie, L. Cao, W. Zhang, K. Sui, Y. Wang, X. Du, G. Duan, and D. Pei, "Actionable and interpretable fault localization for recurring failures in online service systems." New York, NY, USA: Association for Computing Machinery, 2022. [Online]. Available: https://doi.org/10.1145/3540250.3549092

[8] L. Zheng, Z. Chen, J. He, and H. Chen, "Multi-modal causal structure learning and root cause analysis," 2024. [Online]. Available: https://arxiv.org/abs/2402.02357

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

[10] 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*, ser. ESEC/FSE 2023. New York, NY, USA: Association for Computing Machinery, 2023, p. 553–565. [Online]. Available: https://doi.org/10.1145/3611643.3616249

[11] S. Gu, G. Rong, T. Ren, H. Zhang, H. Shen, Y. Yu, X. Li, J. Ouyang, and C. Chen, "Trinityrcl: Multi-granular and code-level root cause localization using multiple types of telemetry data in microservice systems," *IEEE Transactions on Software Engineering*, vol. 49, no. 5, pp. 3071–3088, 2023.

[12] H. Wang, Z. Wu, H. Jiang, Y. Huang, J. Wang, S. Kopru, and T. Xie, "Groot: An event-graph-based approach for root cause analysis in industrial settings," 2021. [Online]. Available: https://arxiv.org/abs/2108.00344

[13] 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)*, 2021, pp. 493–500.

[14] 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.

[15] 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*, ser. CIKM '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 4373–4382. [Online]. Available: https://doi.org/10.1145/3459637.3481903

[16] 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.

[17] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[18] R. Picoreti, A. Pereira do Carmo, F. Mendonça de Queiroz, A. Salles Garcia, R. Frizera Vassallo, and D. Simeonidou, "Multi-level observability in cloud orchestration," in *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech)*, 2018, pp. 776–784.

[19] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11–33, 2004.
[20] R. Rouf, M. Rasolroveicy, M. Litoiu, S. Nagar, P. Mohapatra, P. Gupta, and I. Watts, "Instantops: A joint approach to system failure prediction and root cause identification in microserivces cloud-native applications," in *Proceedings of the 15th ACM/SPEC International Conference on Performance Engineering*, 2024, pp. 119–129.
[21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.
[22] M. Du, F. Li, G. Zheng, and V. 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*, ser. CCS '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 1285–1298.
[23] 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.
[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.
[25] H. Qiu, S. S. Banerjee, S. Jha, Z. T. Kalbarczyk, and R. Iyer, "Preprocessed tracing data for popular microservice benchmarks," 2020.
[26] X. Zhou, X. Peng, T. Xie, J. Sun, C. Xu, C. Ji, and W. Zhao, "Benchmarking microservice systems for software engineering research," in *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings*, 2018, pp. 323–324.
[27] Kubernetes Authors, "Kubernetes: Production-grade container orchestration," https://kubernetes.io, 2023.
[28] Locust Authors, "Locust," https://locust.io.
[29] Prometheus Authors, "Prometheus: Monitoring system & time series database," https://prometheus.io, 2023.
[30] Alibaba, "Chaosblade," https://github.com/chaosblade-io/chaosblade, 2023.
[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017. [Online]. Available: https://arxiv.org/abs/1412.6980
[32] R. Daly, Q. Shen, and S. Aitken, "Learning bayesian networks: approaches and issues," *The Knowledge Engineering Review*, vol. 26, no. 2, p. 99–157, 2011.
[33] OpenTelemetry Authors, "Opentelemetry: High-quality, ubiquitous, and portable telemetry to enable effective observability," https://opentelemetry.io, 2023.
[34] Y. Sui, Y. Zhang, J. Sun, T. Xu, S. Zhang, Z. Li, Y. Sun, F. Guo, J. Shen, Y. Zhang, D. Pei, X. Yang, and L. Yu, "Logkg: Log failure diagnosis through knowledge graph," *IEEE Transactions on Services Computing*, vol. 16, no. 5, pp. 3493–3507, 2023.
[35] B. Sharma, P. Jayachandran, A. Verma, and C. R. Das, "Cloudpd: Problem determination and diagnosis in shared dynamic clouds," in *2013 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2013, pp. 1–12.
[36] D. Y. Yoon, N. Niu, and B. Mozafari, "Dbsherlock: A performance diagnostic tool for transactional databases," in *Proceedings of the 2016 International Conference on Management of Data*, ser. SIGMOD '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 1599–1614. [Online]. Available: https://doi.org/10.1145/2882903.2915218
[37] M. Ma, Z. Yin, S. Zhang, S. Wang, C. Zheng, X. Jiang, H. Hu, C. Luo, Y. Li, N. Qiu, F. Li, C. Chen, and D. Pei, "Diagnosing root causes of intermittent slow queries in cloud databases," *Proc. VLDB Endow.*, vol. 13, no. 8, p. 1176–1189, apr 2020. [Online]. Available: https://doi.org/10.14778/3389133.3389136