Journal of Computer Technology and Software

ISSN:2998-2383

Vol. 3, No. 8, 2024

State-Space Temporal Modeling and Feature Representation Learning for Anomaly Detection in Backend Microservice Systems

Yihan Xue
University of Southern California, Los Angeles, USA
xueyihan2014@gmail.com

Abstract: This paper addresses the challenge of reliably identifying end-to-end latency anomalies in backend microservice architectures by proposing a link-level anomaly detection method based on distributed tracing and multi-source observable signals. This method uses the request-level call chain as the basic object, unifying end-to-end latency, service segment latency, and link structure information. It aligns contextual signals from metrics and logs within a time window to form a joint feature representation usable for detection. To reduce interference from noise, scale differences, and extreme values, robust normalization and lightweight smoothing are employed to normalize the input. Subsequently, neighborhood aggregation is performed based on the service call graph to characterize the consistency relationship between node states and their upstream and downstream components. Residual deviation is used to measure local anomalies and propagational disturbances. Furthermore, end-to-end deviation and node residuals along the path are fused to construct a unified anomaly score, and detection results are output through threshold rules, while retaining link contribution decomposition to support interpretable analysis. Comparative experiments, conducted under a unified evaluation index system, validate the advantages of this method in detection accuracy, recall, overall performance, and discriminative ability, demonstrating that the collaborative modeling of the end-to-end structure and time-series statistics can effectively improve the stability and reliability of microservice latency anomaly detection.

Keywords: End-to-end tracing, multi-source observability, service call graph, and anomaly score fusion

1. Introduction

With the development of cloud-native computing and business platformization, backend systems have gradually evolved into distributed architectures composed of numerous microservices. Services complete a user request through multi-level call chains. The long chain spans, complex dependencies, and dynamically changing operating environments make end-to-end latency a key indicator for evaluating user experience and stability[1]. Compared to monolithic architectures, microservices introduce more uncertainties in terms of elastic scaling, heterogeneous deployment, and cross-regional communication. Any fluctuation in any link can be amplified and ultimately manifest as end-to-end latency anomalies. Therefore, end-to-end anomaly detection related to microservice latency has become one of the core issues for ensuring online business availability and service quality[2].

In real-world scenarios, latency anomalies are often not triggered by a single point of failure, but rather by a combination of factors such as resource contention, dependency degradation, configuration drift, network congestion, and sudden traffic spikes, exhibiting non-stationary and noisy characteristics over time. Simultaneously, the continuous evolution of microservice topologies, frequent instance inactivity and outactivity, and potential changes in call paths such as rerouting and fan-out can easily lead to false positives and false negatives using traditional methods based on static thresholds or single-metric alerts. More importantly, observable anomalies are scattered across multiple data sources, including metrics, logs, and link tracing, leading to difficulties in time alignment, semantic inconsistencies, and incomplete sampling. This makes it challenging to accurately characterize the formation mechanism and propagation process of end-to-end latency anomalies from a purely local perspective[3].

End-to-end anomaly detection requires a systemic approach to model the latency contribution of requests across services and stages, identifying the triggering location, propagation path, and impact range of anomalies, while maintaining robustness and generalization capabilities under complex noise and conceptual drift. This demands that algorithms not only capture cross-service relationships and temporal dependencies but also achieve stable anomaly characterization and rapid localization under conditions of missing data, fluctuating distributions, and the coexistence of heterogeneous signals. Simultaneously, engineering practice emphasizes real-time performance and interpretability; detection results must support rapid decision-making in operations and development, preventing the system from being pushed into alarm storms or mishandling, thus forming a reliable closed-loop governance capability[4].

Therefore, researching end-to-end anomaly detection algorithms for backend microservice latency has significant theoretical and practical value. In theory, it promotes the characterization of cross-component temporal coupling, causal relationships, and anomaly propagation patterns in distributed systems, providing a methodological foundation for the unified representation and fusion of multi-source observable data. In

practice, it helps reduce the cost of fault detection and localization, improve service quality and user experience, reduce business losses and resource waste caused by latency degradation, and provide data-driven support for capacity planning, performance tuning, and risk warning. Building reliable, scalable, and interpretable end-to-end anomaly detection capabilities for complex microservice ecosystems will become a key step in achieving high availability and high resilience for modern backend systems.

2. BackGround

In backend systems designed for high concurrency and rapid iteration, microservices are gradually becoming the mainstream approach for core business logic. While breaking down functionality into finer-grained service units significantly improves development and deployment efficiency, it also makes the system's operational status more difficult to monitor intuitively[5]. A single request often traverses multiple gateways and middleware, undergoing multiple serialization and deserialization processes, thread switching, and queueing, ultimately being completed by several downstream dependencies. Latency is no longer the performance result of a single module, but rather a comprehensive phenomenon resulting from the superposition and mutual constraints of multiple stages, exhibiting strong fluctuations under different business peaks, resource quotas, and scheduling strategies[6].

The causes of latency anomalies are often hidden in subtle changes, such as resource contention caused by local hotspots, additional access due to changes in cache hit rates, intermittent slow responses from dependent services, and the boundary effects of connection pools and rate-limiting strategies. These issues may not be accompanied by obvious error codes or crash signals, but they can continuously increase tail latency and cause user-side lag and timeouts. Meanwhile, the dynamic nature of microservice environments leads to the continuous evolution of anomalies. Adjustments to service versions, runtime parameters, and instance scale can alter the system's baseline state, making methods relying on fixed reference values ineffective in the long run. Anomalies may also exhibit cross-layer linkage characteristics, with multiple factors from infrastructure to the application layer working together, making judgments based on a single perspective prone to bias[7,8].

To address these challenges, the industry has gradually developed a governance approach based on observability, attempting to reconstruct the system's operational context and identify anomalies from multi-source signals. However, at the data level, differences in time granularity and collection frequency across different sources, widespread missing data and noise, and inconsistent semantic mappings increase the difficulty of modeling. At the methodological level, it is necessary to characterize normal patterns and anomaly deviations under complex dependencies and strong temporal correlations, while simultaneously considering the efficiency and interpretability of online detection results, to truly support practical needs such as alarm convergence, root cause analysis, and risk prevention. Therefore, the systematic identification and localization of end-to-end latency anomalies remains a key background issue that urgently requires in-depth research in backend microservice governance.

3. Datasets and Dataset Preprocessing

3.1 Dataset

This paper selects the AIOps Challenge 2021 public dataset as the research object for end-to-end microservice latency anomaly detection. This dataset is designed for anomaly detection and localization scenarios in real-world microservice operating environments, providing three types of observational data: logs, metrics, and call chain tracing. It covers key clues from service call relationships to end-to-end latency changes and is released under an open-source license, facilitating reproduction and comparative research.

In terms of data content, the call chain is organized on a per-request basis, containing multiple span records across services along with their timing and duration information, which can be used to construct service dependency graphs and link-level latency representations. The metrics and logs supplement this with side signals of resource and operational status, supporting multimodal correlation analysis of latency anomalies. Because the data simultaneously possesses link-level latency information and multi-source contextual signals, it has a high degree of matching with the theme of end-to-end anomaly detection for backend microservice latency, making it suitable for observable modeling of anomaly triggering and propagation processes.

3.2 Dataset Preprocessing

Data preprocessing begins with the entire call chain data. Raw trace records are aggregated by trace dimension, and the span sequence of each request undergoes time consistency verification and structural normalization. Specifically, this includes removing records with missing key fields, standardizing timestamp units and time zones, correcting significant negative latency and abnormal spans, and reconstructing and topologically verifying parent-child relationships within the same trace to ensure consistency in call hierarchy, service names, and operation identifiers across the entire data. Subsequently, discrete fields such as services and interfaces are encoded and mapped to form computable node and edge identifiers. Simultaneously, request-level and link-level latency features are constructed, such as end-to-end total latency, the proportion of time spent in each service segment, critical path latency, fan-out count, and cross-service hop count, so that subsequent modeling can directly characterize the distribution and aggregation patterns of latency within the link.

In the metrics and logs section, preprocessing focuses on aligning time sequences, cleaning noise, and performing cross-source correlation. First, multi-source signals are resampled and aligned according to a unified time window. Missing segments are gently filled using methods such as forward padding and local interpolation, and extreme outliers are truncated or robustly smoothed to reduce the interference of acquisition jitter on features. Then, metrics, logs, and call chains are associated with service identifiers and temporal neighborhoods to construct a joint representation of the request chain and its surrounding operational status. Finally, all

continuous features are standardized or robustly normalized, and deduplication and consistency checks are performed at the sample level to ensure stable data distribution and consistent field semantics during training and evaluation, thus providing high-quality input for end-to-end latency anomaly detection. Figure 1 shows a comparison of the dataset before and after preprocessing.

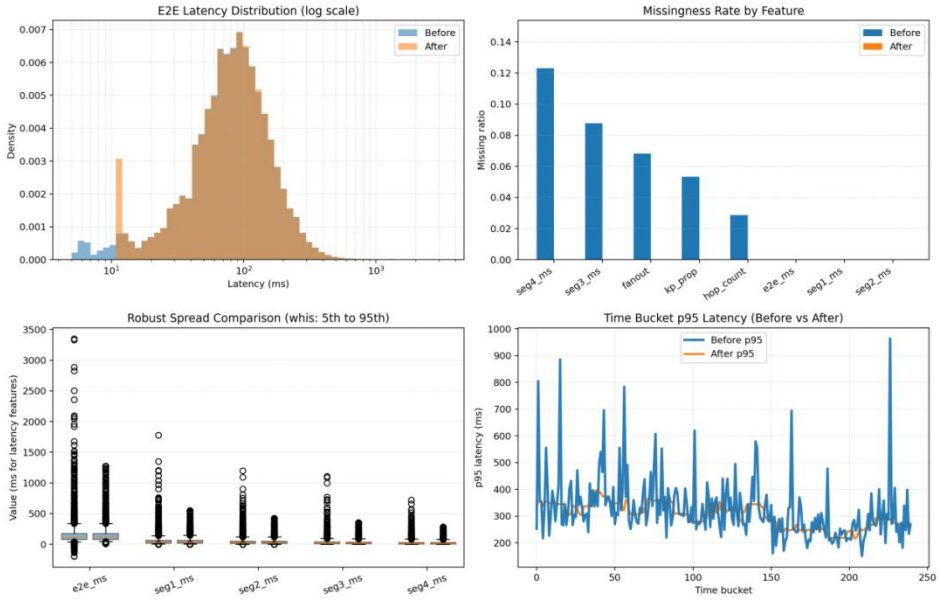

Figure 1. Comparison of experimental results before and after data preprocessing

3. Method

End-to-end anomaly detection for backend microservice latency requires simultaneously characterizing the propagation structure of requests along the call chain and their performance state over time. This paper's method uses the tracing chain of a single request as the basic object, representing it as a directed path composed of multiple service segments. It aligns metrics and chain latency within a unified time window, constructing a feature representation that considers both structure and timing. Specifically, it first aggregates chain-level features such as end-to-end latency and critical path percentage for each trace. Simultaneously, it statistically analyzes structural quantities such as service segment latency, fan-out, and hop count at the span level, mapping these features to service nodes and inter-service call edges to form a call graph signal that evolves. Considering the inherent load fluctuations and heterogeneous scale differences in microservice scenarios, the method performs robust standardization and lightweight smoothing before entering the detection module. This ensures that subsequent anomaly scores reflect relative deviations rather than absolute magnitude differences, thereby improving comparability and consistency across different services and time periods. This article also provides an overall model architecture diagram, as shown in Figure 2.

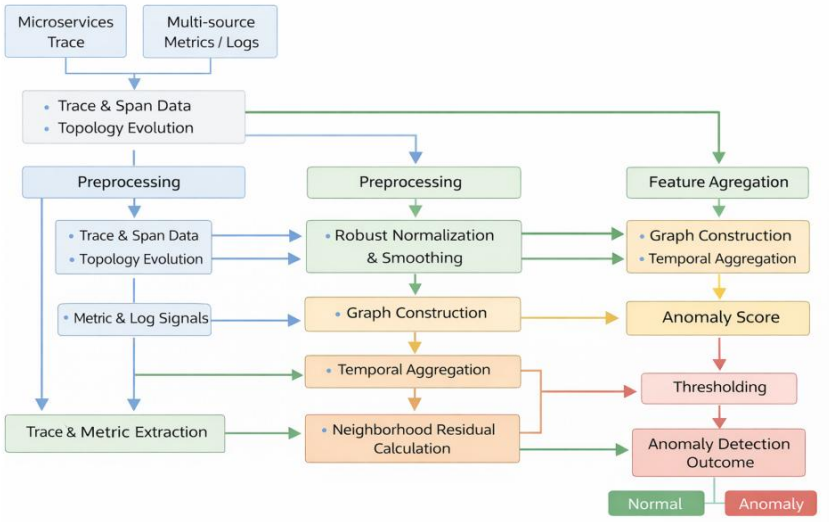

Figure 2. Overall model architecture diagram

To unify the dimensions of different features and reduce the impact of extreme values, robust normalization based on the median and interquartile range is applied to each feature. For any observation x, its standardized result x^* is defined as:

$$x^* = \frac{x - median(x)}{IQR(x) + \epsilon}$$

Where $IQR(x)$ is the interquartile range and ϵ is a minimal constant to avoid division by zero. Subsequently, an exponential moving average is applied to the key link-level statistics over time to suppress jitter. Let the smoothed value at time t be μ_t, then:

$$\mu_t = \alpha x_t + (1-\alpha)\mu_{t-1}$$

Where x_t represents the observation statistics of the time bucket, and $\alpha \in (0,1)$ controls the smoothing intensity. Through the above processing, multi-source signals can be transformed into more stable sequence inputs, providing a reliable basis for subsequent structure aggregation and deviation calculation.

In structural modeling, the method constructs a service call graph from the call chain and uses an adjacency matrix to perform lightweight aggregation of inter-service influences. Let the service graph have n service nodes, and the adjacency matrix be $A \in R^{n \times n}$, and the eigenvector of node i at time t be $h_i^{(t)}$. Then, its neighborhood aggregation representation is defined as:

$$g_i^{(t)} = \sum_{j=1}^{n} A_{ij} h_j^{(t)}$$

This formula incorporates the state changes of upstream and downstream into the same representation, thus reflecting the interconnectivity of the entire link. Subsequently, the abnormal deviation of the node state is measured in residual form, and the residual of node i is defined as:

$$r_i^{(t)} = h_i^{(t)} - g_i^{(t)}$$

The larger the absolute magnitude of the residual, the weaker the consistency between the current state of the service and its neighborhood, and the more likely it is to have local anomalies that spread to the link.

Finally, the method combines node-level residuals and link-level end-to-end deviations into a simple anomaly score. Let the end-to-end delay of a request at time t be L_t, and its smoothing reference is μ_t; then the link deviation term is $|L_t - \mu_t|$. Let the set of services traversed by this request be P; then the comprehensive anomaly score is defined as:

$$S_t = |L_t - \mu_t| + \lambda \sum_{i \in P} \|r_i^{(t)}\|_1$$

Where λ is used to balance the contribution of link deviation and node residual, and $\|\cdot\|_1$ is the sum of the absolute values of the elements. When making a judgment based on the score, a threshold τ is introduced to give the final label.

$$y_t = \begin{cases} 1, S_t > \tau \\ 0, S_t \le \tau \end{cases}$$

Here, $y_t = 1$ indicates that an abnormal end-to-end latency was detected, and $y_t = 0$ indicates that it is normal. This judgment rule is simple in form, has low computational overhead, and the score can be decomposed into the sum of end-to-end deviation and the contribution of each service on the path, which facilitates the formation of interpretable anomaly localization clues.

4. Experimental Results and Analysis

4.1 Experimental setup

This study employed a single-machine, single-GPU configuration for both training and inference. Hardware-wise, each compute node was equipped with an NVIDIA RTX 4090 GPU with 24GB of VRAM for model training and batch inference; a 16-core CPU supported data loading and feature construction; 64GB of memory was allocated to meet the requirements of multi-source time-series data alignment, caching, and batch processing; and NVMe SSDs were used for storage to accelerate random read/write operations on log and tracking data. Software-wise, the operating system was Ubuntu 20.04 LTS, the development language was Python 3.10, the deep learning framework was PyTorch 2.2, and CUDA 12.1 and cuDNN were used for GPU acceleration. Data processing and statistical analysis primarily relied on commonly used components such as NumPy, Pandas, and Matplotlib, ensuring the experimental workflow was reproducible and easy to deploy and migrate.

For hyperparameter settings, to balance training stability and convergence speed, the AdamW optimizer was used with a learning rate of 1e-3, weight decay of 1e-4, and batch size of 256. Gradient clipping was used to limit the global norm to 1.0 to avoid gradient explosion. The number of training epochs was set to 80, and a 5-epoch linear warmup and cosine annealing learning rate scheduling were employed to improve optimization stability in the later stages. On the input side, a fixed time window was used to align the links and metrics, with a time bucket granularity of 1 minute and a sliding step size of 1 minute. Exponential moving average was used for temporal smoothing, with a smoothing coefficient α set to 0.2. The anomaly detection threshold was set using a high-quantile strategy based on the training set score distribution, with the default value corresponding to the 0.995 quantile as τ. To reduce the impact of randomness, all experiments used a fixed random seed of 42, and the data partitioning and preprocessing procedures were kept consistent.

4.2 Experimental Results and Analysis

To verify the effectiveness and comparability of the proposed method in the microservice end-to-end latency anomaly detection task, this paper selects several representative works related to distributed tracing and multi-source observable data analysis as comparative baselines, and conducts quantitative comparisons under a unified evaluation index system to reflect the differences in detection accuracy and stability of different methods. The experimental results are shown in Table 1.

Table 1: Experimental results compared with other models

Method	Precision	Recall	F1-score	AUROC
Panahandeh et al.[9]	0.72	0.68	0.70	0.81
Chen et al.[10]	0.75	0.70	0.72	0.84
Xie et al.[11]	0.78	0.73	0.75	0.86
Xie et al.[12]	0.80	0.74	0.77	0.87
Kohyarnejadfard et al.[13]	0.82	0.76	0.79	0.89
Zhang et al.[14]	0.83	0.78	0.80	0.90
Li et al.[15]	0.85	0.80	0.82	0.92
Ours	0.91	0.88	0.89	0.97

Overall, the comparative methods show a gradual improvement in the balance between precision and recall. Models closer to the later stages are better able to simultaneously reduce false positives and false negatives, which is directly reflected in the simultaneous improvement of comprehensive metrics. This indicates that as methods move from more general anomaly detection to modeling more closely aligned with the semantics of microservice links, the models more clearly characterize the boundaries of latency anomalies, identifying genuine anomalies without overreacting to normal fluctuations.

Comparing the middle tier reveals that some methods excel in certain metrics, but their overall performance doesn't show a significant difference, reflecting their continued susceptibility to link noise, local jitter, and dependency changes. Particularly when there are inconsistencies in multi-source signals or dynamic topology changes at the link level, strategies relying solely on a single perspective or weak structure modeling are more prone to judgment inconsistencies, making it difficult to maintain high precision and recall simultaneously, thus limiting overall stability.

The proposed method maintains a more consistent advantage across all four evaluation dimensions, especially with a more significant improvement in comprehensive metrics and overall discriminative power, demonstrating that modeling from a full-link perspective indeed brings stronger discriminative power. This advantage is more like a synergy of three stages: robust preprocessing reduces the interference of noise and scale differences, structural aggregation makes it easier to capture anomalies and deviations from neighborhood consistency, and finally, a unified scoring mechanism combines end-to-end deviations and path contributions, making the judgment more stable, more interpretable, and more suitable for cross-component cumulative problems such as microservice latency.

The smoothing coefficient directly affects the jitter suppression strength of time-series signals, thereby altering the stability and boundary clarity of outlier scores. To evaluate the robustness of the model under different smoothing intensities, it is necessary to observe the distribution pattern of key discriminative capabilities as parameters change. This experiment uses graded perturbations on the smoothing coefficient and statistically analyzes the performance fluctuations under multiple repeated samplings to characterize the stable and sensitive regions. The experimental results are shown in Figure 3.

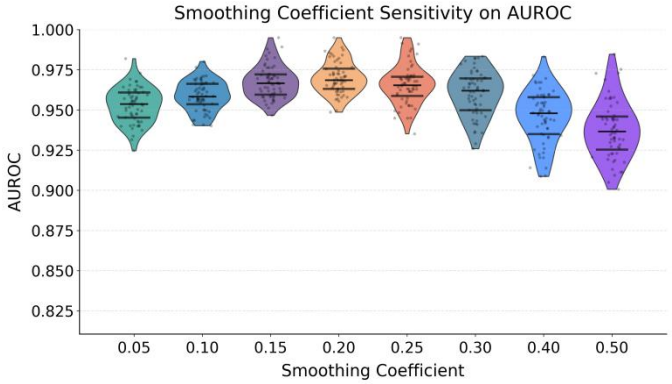

Figure 3. Smoothing coefficient sensitivity experiment to AUROC

From a graphical perspective, AUROC exhibits a characteristic of initial stabilization followed by a decline as the smoothing coefficient changes. The more concentrated distribution in the middle segment indicates a good balance between suppressing temporal noise and preserving anomalous boundaries within this range, resulting in more stable overall discrimination ability. The more divergent distribution at both ends suggests that excessively small or large parameters amplify uncertainty, making the same input perturbation more likely to cause fluctuations in the scoring system.

Further analysis of the width and median position of each violin group reveals that weak smoothing allows high-frequency jitter to more easily enter the score calculation, amplifying the tail behavior of different time buckets, manifesting as a wider distribution and longer tails. Conversely, when smoothing is too strong, local peaks and short-term abrupt changes are excessively suppressed, anomalous signals are flattened, and the model's distinction of boundary samples becomes more dependent on the overall trend, resulting in an overall downward shift and increased dispersion. This change aligns with the role of robust normalization and sliding smoothing emphasized in the previous method design; parameter selection directly determines the trade-off between noise suppression and fidelity preservation.

From an engineering perspective, this figure provides a relatively clear usable range, minimizing random fluctuations without sacrificing anomaly visibility. Echoing the overall advantages demonstrated in previous comparative experiments, the sensitivity results here indicate that the method does not rely on a single extreme parameter to perform well, but rather possesses transferable stability within a reasonable range. It also suggests that under different workloads and sampling qualities, small-scale parameter tuning can be performed

around this stable range without significantly altering the threshold or feature construction strategy.

Perturbations in the proportion of missing values alter the integrity of the observable signal, thus affecting how outlier scores characterize key patterns. To test the robustness of the method under incomplete data conditions, the degree of missing values can be gradually increased, and the trend of discriminative ability can be observed. This experiment simulates data quality fluctuations that more closely resemble real-world acquisition scenarios by applying different intensities of missing value perturbations to the original observations.

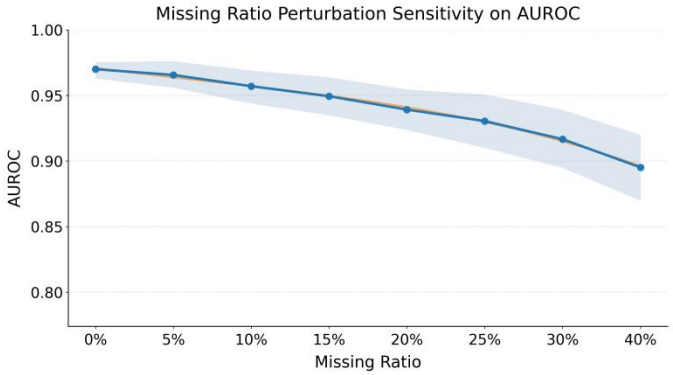

Figure 4. Sensitivity experiment of missing value proportion perturbation to AUROC

As shown in Figure 4, the AUROC score gradually decreases with increasing missing perturbation, which is consistent with the intuitive understanding that the anomaly boundary information is weakened as the observable signal becomes sparser. The initial decline is relatively gradual, indicating that the method has some tolerance for mild to moderate missing data and can still extract stable clues from the preserved link structure and time-series statistics. As the missing data increases, the curve declines more significantly, indicating that the discriminative power of the anomaly score is more directly affected when key signals are heavily obscured.

The light blue band gradually widens as the missing data worsens, reflecting increased volatility in the results. This echoes the conclusions of the previous smoothing coefficient sensitivity experiment, but the mechanism is different. There, the effect was more on the noise suppression intensity's impact on boundary preservation, while here the core issue is the reduction in the amount of information itself, leading to greater differences in usable evidence for the same link under different missing data samples. In other words, missing data not only reduces the average discriminative power but also worsens stability, making the model more susceptible to the influence of accidental missing data locations.

From a methodological design perspective, this trend demonstrates that robust normalization and link aggregation do indeed provide fundamental support, enabling relatively reliable discrimination capabilities even when data loss is not severe. However, as data loss worsens, the joint score of local residuals and end-to-end deviations faces the problem of insufficient input, especially since missing service segment information on critical paths weakens the capture of propagation anomalies. Therefore, this experiment is more like characterizing a usable data quality boundary, suggesting that ensuring coverage of key tracking fields and core indicators on the data collection side is crucial to continuously leveraging the advantages of end-to-end modeling.

5. Conclusion

Addressing real-world challenges such as long call chains, complex dependencies, strong signal-noise imbalances, and heterogeneous data sources, it proposes a detection approach based on multi-source observable data and oriented towards the link propagation mechanism. The method uses request-level tracing as its core, unifying the representation of end-to-end latency, service segment latency, topological relationships, and temporal evolution information. Robust preprocessing enhances comparability across services and time periods, and structural aggregation and residual deviation characterize link consistency changes, ultimately forming an interpretable anomaly scoring and judgment mechanism. The overall framework emphasizes lightweight and practicality, adapting to the real-time requirements of online systems and seamlessly integrating with existing observability systems in engineering, providing a more end-to-end analytical tool for microservice governance.

Comparative experiments show that the proposed method exhibits more consistent advantages across multiple evaluation dimensions, demonstrating comprehensive improvements in false positive control, false negative suppression, and overall discrimination capabilities. More importantly, this improvement does not rely on a single signal or a single-point assumption, but rather stems from the collaborative modeling of link structure and timing statistics. This allows the model to more accurately grasp the impact of anomaly propagation on end-to-end latency under complex dependencies. This conclusion implies that when dealing with latency-related problems, it is necessary to place the local service state back into the call chain context for joint judgment. This lays the foundation for building a more stable and interpretable anomaly detection and diagnosis closed loop, and provides methodological support for related systems to move from passive alerting to proactive governance.

At the application level, this research has direct value for highly latency-sensitive fields such as cloud-native platforms, online service systems, financial transaction and payment gateways, real-time recommendation and advertising, industrial internet, and edge computing. End-to-end latency anomalies often lead to decreased user experience, transaction timeouts, cascading degradation, and resource waste, even causing business losses and compliance risks. The method presented in this paper can identify link-level anomaly signs earlier and more reliably, and provide location clues for operations and development through path contribution decomposition, helping to shorten troubleshooting time, reduce alarm noise, and improve service stability. Meanwhile, in scenarios such as capacity planning, change assessment, and performance regression, this method can also serve as a continuous monitoring component, providing quantitative risk alerts for

system iterations, thereby driving the evolution of microservice platforms towards high availability and high resilience.

Future work can be further deepened by starting with broader production constraints and more complex system evolution. On the one hand, anomaly detection can be more closely linked with root cause localization, impact assessment, and automated handling strategies, enabling detection results to be naturally transformed into actionable governance actions and achieving more granular strategy switching under different business objectives. On the other hand, as microservice topologies and deployment patterns continue to change, maintaining long-term robustness under dynamic topology evolution, missing observation signals, and distribution drift remains a key direction, and stronger adaptive representation and continuous learning mechanisms can be explored. At the same time, cross-domain data isolation, privacy constraints, and resource limitations brought about by multi-cloud and edge scenarios also pose new challenges. Future research can further explore lightweight and privacy-friendly end-to-end modeling methods to promote the implementation and industrial application of such methods in larger-scale and more complex ecosystems.

References

[1] Zhang C, Peng X, Sha C, et al. Deeptralog: Trace-log combined microservice anomaly detection through graph-based deep learning[C]//Proceedings of the 44th international conference on software engineering. 2022: 623-634.

[2] Lee C, Yang T, Chen Z, et al. Eadro: An end-to-end troubleshooting framework for microservices on multi-source data[C]//2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE). IEEE, 2023: 1750-1762.

[3] Zhang C, Dong Z, Peng X, et al. Trace-based multi-dimensional root cause localization of performance issues in microservice systems[C]//Proceedings of the IEEE/ACM 46th International Conference on Software Engineering. 2024: 1-12.

[4] Zhang Z, Ramanathan M K, Raj P, et al. {CRISP}: Critical path analysis of {Large-Scale} microservice architectures[C]//2022 USENIX Annual Technical Conference (USENIX ATC 22). 2022: 655-672.

[5] Raeiszadeh M, Ebrahimzadeh A, Saleem A, et al. Real-time anomaly detection using distributed tracing in microservice cloud applications[C]//2023 IEEE 12th International Conference on Cloud Networking (CloudNet). IEEE, 2023: 36-44.

[6] Colarusso C, De Caro A, Falco I, et al. A distributed tracing pipeline for improving locality awareness of microservices applications[J]. Software: Practice and Experience, 2024, 54(6): 1118-1140.

[7] C. Wen, "Modeling Evolving Service Dependencies: Dynamic Graph Learning for Microservice Anomaly Detection", 2024.

[8] Z. Li, "Log Event Graph Modeling for Backend Anomaly Detection with Multi-Relational Representation Learning", 2024.

[9] Panahandeh M, Hamou-Lhadj A, Hamdaqa M, et al. ServiceAnomaly: An anomaly detection approach in microservices using distributed traces and profiling metrics[J]. Journal of Systems and Software, 2024, 209: 111917.

[10] Chen J, Liu F, Jiang J, et al. TraceGra: A trace-based anomaly detection for microservice using graph deep learning[J]. Computer Communications, 2023, 204: 109-117.

[11] Xie Z, Xu H, Chen W, et al. Unsupervised anomaly detection on microservice traces through graph vae[C]//Proceedings of the ACM Web Conference 2023. 2023: 2874-2884.

[12] Xie Z, Pei C, Li W, et al. From point-wise to group-wise: A fast and accurate microservice trace anomaly detection approach[C]//Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 2023: 1739-1749.

[13] Kohyarnejadfard I, Aloise D, Azhari S V, et al. Anomaly detection in microservice environments using distributed tracing data analysis and NLP[J]. Journal of Cloud Computing, 2022, 11(1): 25.

[14] C. Xu, "Intelligent Defect Detection and Risk Assessment for Cloud Platforms Using Counterfactual System Modeling", 2024.

[15] Li R, Du M, Wang Z, et al. Longtale: Toward automatic performance anomaly explanation in microservices[C]//Proceedings of the 2022 ACM/SPEC on International Conference on Performance Engineering. 2022: 5-16.