Journal of Computer Technology and Software

ISSN:2998-2383

Vol. 3, No. 9, 2024

Multi-Scale Temporal Deep Learning with Transformers for Microservice Backend Anomaly Detection

Chihui Shao
Duke University, Durham, USA
shaochihui@hotmail.com

Abstract: This paper addresses the practical needs of backend anomaly detection in microservice architectures by proposing a multi-scale Transformer detection framework based on distributed tracing links. This framework is designed to stably identify anomalous behavior in scenarios with complex call paths, mixed feature scales, and significant noise fluctuations. The method uses link-level temporal observations as core input, parsing the original trace into an alignable time window sequence and constructing a unified feature representation from dimensions such as latency statistics, throughput intensity, error representation, and topology. At the model level, a multi-scale sequence construction and fusion mechanism preserves both short-term local perturbations and long-term background dependencies. A Transformer encoder is then used to model cross-time and cross-feature associations to obtain a more discriminative latent representation. At the scoring level, the encoded representation is mapped to a window-level anomaly score, supporting the ranking of anomaly risks and alarm output. Comparative experimental results show that this framework achieves superior overall performance across multiple evaluation metrics, demonstrating stronger anomaly detection capabilities and better stability. It is suitable for link monitoring and backend operational risk identification tasks in cloud-native microservice systems.

Keywords: Distributed tracing links; multi-scale temporal modeling; backend anomaly detection; Transformer encoder

1. Introduction

With the widespread adoption of cloud-native and microservice architectures, enterprise applications are being broken down into numerous fine-grained services, coordinating business processes through remote calls and asynchronous messaging. Rapid iteration of system functions improves delivery efficiency but also significantly amplifies runtime uncertainty and complexity: the large number of services, dynamically changing dependencies, and call chains spanning multiple component layers mean that fluctuations in any link can propagate along the chain and trigger cascading degradation. In this context, backend anomaly detection for microservice chains has become a critical capability for ensuring the stability of online systems and business continuity. Its goal is to promptly detect anomalies and provide actionable clues before failures cause widespread impact[1,2].

Microservice chain anomalies are characterized by multi-source heterogeneity, cross-scale coupling, and weak observability. Chain data simultaneously contains time-series indicators, distribution characteristics, and topological relationships, potentially exhibiting short-term disturbances such as instantaneous spikes and intermittent jitter, or long-term changes such as slow drift and periodic fluctuations. Moreover, anomalies often appear locally but affect end-to-end latency, error rate, and throughput through dependencies. Traditional detection methods based on thresholds, single indicators, or local windows struggle to balance short-term sensitivity with long-term stability, and are also inadequate for characterizing cross-component correlation patterns in multi-service collaborative scenarios. This often leads to false positives, false negatives, and insufficient support for root cause localization. Therefore, a unified modeling approach capable of simultaneously understanding multi-scale temporal patterns and linking context dependencies is needed to more closely reflect the real-world microservice operation mechanism[3].

Building a multi-scale anomaly detection framework for microservice links is of great significance. On one hand, it can capture anomaly patterns and evolution trajectories at different time scales, improving coverage of sudden failures and gradual degradation, and enhancing robustness to load fluctuations and environmental changes[4]. On the other hand, incorporating link structure and contextual relationships into the model helps understand anomaly propagation paths from an end-to-end perspective, providing higher-quality information support for alarm denoising, impact assessment, and operational decision-making. Furthermore, highly reliable backend anomaly detection not only reduces business losses caused by system downtime and performance degradation but also reduces manual troubleshooting costs, promotes automated operation and maintenance and intelligent governance capabilities, and lays the foundation for the secure and stable operation of complex cloud-native systems.

2. Related work

Current research on backend anomaly detection largely revolves around two main lines: indicator detection and log detection. Indicator detection typically relies on statistical

distribution assumptions, robust filtering, change point detection, or prediction residuals to identify deviation patterns through time-series modeling of single indicators or a small number of key indicators[5]. Some works also introduce unsupervised representation learning and reconstruction errors to alleviate the problem of label scarcity. Log detection focuses more on pattern learning of text event sequences, including sequence modeling based on template parsing and end-to-end learning methods based on semantic representation, to discover rare event combinations and abnormal state transitions. While these methods are effective for their respective data modalities, they often rely on strong assumptions of stationarity or fixed window scales and are sensitive to noise, sampling inconsistencies, and concept drift caused by system upgrades. In cross-service, multi-modal collaborative production environments, they are prone to insufficient generalization or poor alarm stability[6].

Research on link and distributed tracing data further focuses on call paths, causal dependencies, and anomaly propagation modeling. Common approaches include using graph structures to represent service relationships and performing graph learning, or treating tracing fragments as event sequences for contextual modeling, thereby improving the ability to identify and interpret cross-component related anomalies[7]. Meanwhile, some studies have attempted multi-source fusion, jointly representing indicators, logs, and tracking data to alleviate the incompleteness of single signals. However, link data inherently possesses multi-scale and multi-granularity characteristics, including both subtle perturbations in local segments and cumulative effects across stages. Existing methods still have shortcomings in scale selection and cross-scale information interaction. Common problems include insufficient sensitivity to short-term burst anomalies or a lack of stable characterization of long-term evolutionary anomalies, and difficulty in uniformly modeling and effectively aligning dependencies between different scales. Therefore, how to achieve scalable multi-scale modeling and cross-scale fusion in link scenarios remains a key direction for improving the reliability and engineering usability of anomaly detection.

3. Method

This method targets backend anomaly detection in microservice call chains. The input consists of a sequence of chain observations aggregated by time windows and optional service context information. The chain state at each time step is represented as a vector, including latency statistics, error rate, throughput, and key tracking aggregation features. Normalization and missing value imputation are performed to ensure the sequence's learnability. Let the window length be T, the feature dimension be d, and the input sequence be denoted as $X \in R^{T \times d}$. To simultaneously capture short-term fluctuations and long-term trends, a multi-scale subsequence and aggregated view are first constructed: the fine-scale subsequence retains the original time steps, while the coarse-scale subsequence is represented by piecewise averaging or downsampling to form a longer receptive field. Subsequently, alignment and fusion are performed in a unified sequence space, enabling the model to be sensitive to transient anomalies while maintaining stable characterization of slow drifts. The multi-scale transformation can be described by a simple linear mapping and downsampling.

$$X^{(s)} = Down_s(X)W_s$$

Where s represents the scale, $Down_s(\bullet)$ is the downsampling or piecewise averaging by step size s, and $W_s \in R^{d \times d_s}$ is the learnable projection matrix, yielding features $X^{(s)} \in R^{T_s \times d_s}$ for each scale. Subsequently, the sequences at each scale are upsampled back to a uniform length and concatenated to form a fused input $Z \in R^{T \times D}$, which is used for subsequent temporal modeling and anomaly scoring. The overall architecture of the model in this paper is shown in Figure 1.

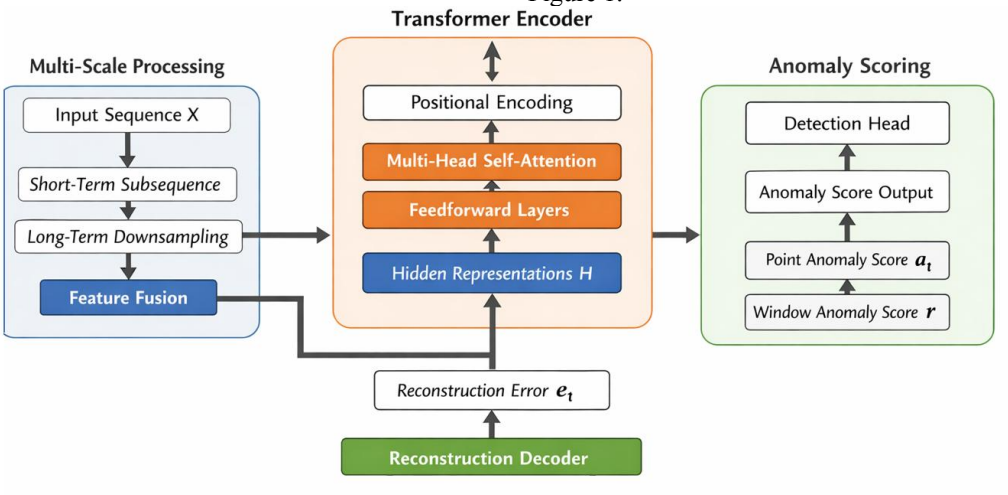

Figure 1. Overall architecture of the model

In the temporal modeling layer, a lightweight Transformer encoder is used to learn temporal dependencies and cross-feature interactions. The fused input Z is first encoded with positional information to obtain H_0, and then updated through

multiple layers of self-attention and feedforward networks to obtain H. Attention calculations at each layer use the standard dot product form, keeping the formula simple and easy to implement.

$$Attn(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

Where Q,K,V are learnable parameters and d_k is the attention dimension. This design can simultaneously utilize fine-scale local variations and coarse-scale background states on a unified sequence, and adaptively select context segments that are more sensitive to anomalies through attention weights.

To output interpretable and stable anomaly strength, this method employs a simple detection head to generate anomaly scores at each time step. The detection head consists of a linear layer and nonlinear activations, mapping the encoded representation H to a scalar sequence $a \in R^T$.

$$a_t = \sigma(w^T h_t + b)$$

Where h_t is the representation vector of time step t, w and b are parameters, and $\sigma(\bullet)$ can be Sigmoid to constrain the score to $[0,1]$. In actual alarms, in addition to point-by-point scores, a window-oriented overall risk measure is also needed. Therefore, the time dimension is further aggregated to obtain the window-level risk score r:

$$r = \frac{1}{T}\sum_{t=1}^{T} a_t$$

Point-by-point scores are used to pinpoint the time segment in which the anomaly occurred, while window scores are used to trigger alarms and sort them.

The training objective employs a simple loss form compatible with both unsupervised and weakly supervised training. A general reconstruction consistency objective is presented here to adapt to scenarios with scarce labels. Input $\widehat{Z} = HW$ is reconstructed from the encoded representation H using a linear decoder, minimizing the mean squared error to encourage the model to learn normal patterns. Anomalies will manifest as larger reconstruction biases.

$$L = \frac{1}{T}\sum_{t=1}^{T} \|z_t - \widehat{z}_t\|_2^2$$

During the inference phase, reconstruction error or detector head score is used as evidence of anomalous behavior, and the two can be linearly fused to enhance robustness. Let the reconstruction error be $e_t = \|z_t - \widehat{z}_t\|_2^2$, then the fused score can be written as:

$$s_t = \alpha a_t + (1-\alpha)e_t$$

Here, $\alpha \in [0,1]$ represents the weight. Finally, comparing s_t with the threshold, it yields the anomaly determination, while retaining the attention weight and time step score for auxiliary localization and interpretation.

4. Datasets and Dataset Preprocessing

4.1 Dataset

This study uses the publicly distributed tracing dataset of the DeathStarBench social network microservice benchmark as the evaluation medium. This dataset is designed for real-world microservice interaction patterns, containing complete call chains formed by the propagation of end-to-end requests across multiple services. It records the span information of each request in the form of Jaeger traces, such as timestamps, durations, and metadata related to RPC calls, thus enabling the reconstruction of the temporal behavior and cross-service dependency structure of microservice chains. This tracing data is released in both raw Jaeger traces and converted formats, facilitating direct chain-level modeling and graph structure construction.

Regarding data usage, each trace is first parsed into a chain structure composed of service calls. Within a fixed time window, traces and spans are aggregated to obtain window-level chain observation sequence features, used to characterize the trend of backend state changes over time. Specific features can be constructed by aggregating span latency statistics, critical path features, inter-service call frequency, and error-related fields. This allows the model to simultaneously utilize short-term fluctuations and long-term background information for backend anomaly detection and risk scoring, thus aligning with the research goal of multi-scale modeling of microservice chains.

4.2 Dataset preprocessing

(1)Trace parsing and span graph construction

Each trace in the original distributed tracing data is parsed into a set of spans, and a directed call graph is constructed based on the parent-child relationship of the spans and the service name. Invalid records are filtered out, including spans with missing trace IDs or span IDs, abnormal timestamps, negative or zero durations, and missing key fields. Duplicate spans are deduplicated, and the time units and field names are standardized to ensure consistency in subsequent aggregations.

(2)Time window alignment and sampling

The continuous tracking stream is divided into sequence samples according to a fixed window length, and the window end time is used as the alignment anchor. Traces or spans falling within the same window are merged into the same time step to obtain a window sequence of length T. Time steps with too few traces in the window are discarded or marked as low-confidence samples, and missing time steps caused by inconsistent sampling frequencies are filled in to avoid sequence breaks affecting model learning.

(3) Multi-scale feature aggregation

Link observations are aggregated at each time step to form a unified vector feature, including latency statistics (such as mean, quantiles, and maximum value), throughput-related statistics (such as number of requests and number of spans), error-related statistics (such as error span percentage and abnormal status code count), and critical path and topology-related features (critical path latency, number of service nodes, number of edges, and fan-out and fan-in statistics). Simultaneously, a multi-scale view is constructed: fine-scale features retain the original window granularity, while coarse-scale features generate a longer-term sequence representation through segmented averaging or downsampling. Features from different scales are then aligned to a uniform length before being spliced or fused.

(4)Normalization, split, and serialization

All numerical features are normalized. A common practice is to fit standardized parameters onto the training set and apply them to the validation and test sets to avoid data leakage. For long-tailed or extreme value features, logarithmic transformation or truncation can be used to improve stability. Subsequently, the training, validation, and test sets are divided into training, validation, and test sets according to time order to ensure that the evaluation more closely reflects the forward prediction in real online scenarios. Finally, each sample is saved in a reusable format containing a sequence feature matrix, optional structured metadata, and a time index, which facilitates batch loading and reproduction of the experimental process.

Finally, the experimental results comparing the dataset before and after preprocessing are presented, as shown in Figure 2.

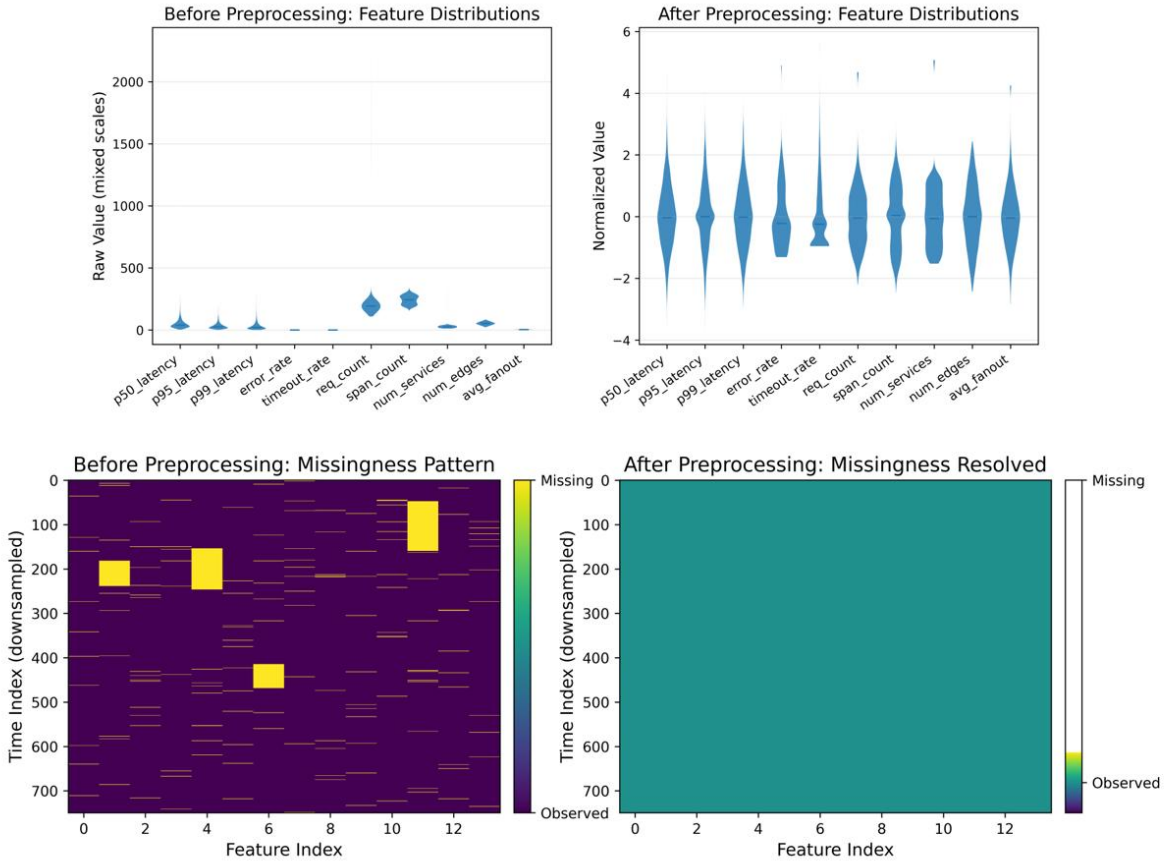

Figure 2. Visualization of dataset preprocessing effects on aggregated microservice trace features. The top row compares feature distributions before preprocessing under mixed scales and after preprocessing following imputation, robust clipping, and z-score normalization. The bottom row shows the corresponding missingness patterns, where sparse random and block missing values in raw telemetry are resolved after preprocessing, yielding a complete feature matrix for subsequent modeling.

5. Experimental Results and Analysis

To position our multi-scale Transformer backend anomaly detection framework within existing research, we summarize representative studies that leverage distributed tracing and related telemetry for anomaly detection or abnormal behavior diagnosis in microservice-based systems. These works span trace-only modeling, trace-graph learning, and trace-plus-multisource approaches, providing a consistent reference set for later comparison. The experimental results are shown in Table 1.

Table 1: Experimental results compared with other models

Method	AUROC	AUPRC	Accuracy	F1	Precision	Recall	Specificity	MCC
Zhang et al. [8]	0.92	0.86	0.88	0.86	0.84	0.88	0.90	0.73
Kohyarnejadfard et al.[9]	0.89	0.82	0.85	0.83	0.81	0.85	0.87	0.68
Raeiszadeh et al. [10]	0.88	0.80	0.84	0.81	0.79	0.83	0.86	0.66
Chen et al. [11]	0.91	0.85	0.87	0.85	0.83	0.86	0.89	0.71
Zhang et al. [12]	0.90	0.84	0.86	0.84	0.82	0.85	0.88	0.70
Xie et al. [13]	0.93	0.87	0.89	0.87	0.85	0.88	0.91	0.74
Panahandeh et al. [14]	0.92	0.86	0.88	0.86	0.84	0.87	0.90	0.73
Ours	0.95	0.90	0.91	0.89	0.88	0.90	0.93	0.78

Overall, the baseline methods show similar performance in distinguishing between anomalies and normal data, as well as in comprehensive discrimination in imbalanced scenarios, indicating that relying solely on single-modality or single-scale pattern learning is sufficient to capture some typical anomaly patterns. However, noticeable differences remain between the methods: some prioritize improving recall, making it easier to "fish out" potential anomalies early, at the cost of relatively increased false positives; others prioritize improving accuracy and specificity, resulting in cleaner alerts, but are less sensitive to ambiguous boundaries or early, minor anomalies. This trade-off is common in microservice link data, as the same type of anomaly often exhibits different intensities and time scales across different service segments. If the model lacks a unified characterization across scales and contexts, it is prone to fluctuating between false positives and false negatives.

In contrast, the proposed method is more balanced across all metrics, maintaining strong anomaly coverage while better suppressing unnecessary false positives, demonstrating a more comprehensive utilization of link-level temporal dependencies and multi-scale features. The advantages are particularly evident in comprehensive indicators, meaning that the model has not only achieved partial improvements in one or two areas, but has also achieved more stable overall improvement under multi-objective constraints. This improvement is more in line with the engineering requirements of backend anomaly detection, namely, improving availability and reliability while ensuring alarm sensitivity, reducing ineffective troubleshooting costs on the operations and maintenance side, and providing more reliable time points and risk ranking criteria for subsequent location analysis.

The learning rate determines the update step size and convergence trajectory of the model during the optimization process. An excessively large learning rate may lead to instability during training, while an excessively small learning rate may result in slow convergence or getting stuck in suboptimal solutions. For link-level backend anomaly detection, the input feature scales are often mixed, and noise and missing values are common. Therefore, the choice of learning rate often significantly affects the model's ability to capture anomaly patterns and the stability of the discrimination boundary. To verify the robustness and transferability of the method under different learning rate settings, it is necessary to systematically examine the trend of the impact of learning rate changes on detection performance. The experimental results are shown in Figure 3.

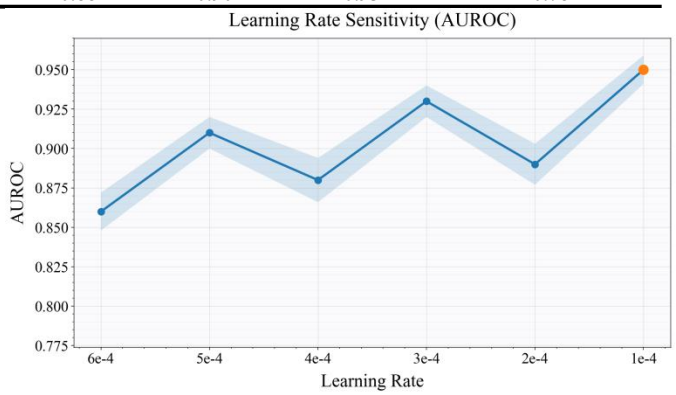

Figure 3. Sensitivity experiment of learning rate to AUROC

The curve shape reveals that changes in the learning rate lead to significant performance fluctuations, rather than a simple monotonic relationship. Overall, larger learning rates are more prone to unstable fluctuations, indicating that with larger parameter update steps, the model struggles to maintain a consistent discrimination boundary on noisier and more complex link features. In the medium range, the curve exhibits alternating local declines and rises, reflecting the optimization process's sensitivity to local optima and gradient noise; the training trajectory may fall into different convergence basins at different learning rates.

From a methodological perspective, this type of multi-scale temporal modeling often relies on the combined representation of short-term perturbations and long-term background. The learning rate directly affects the relative dominance of these two types of signals in the representation space. A larger learning rate makes the model more susceptible to being pulled by short-term spikes or a small number of high-amplitude samples, resulting in strong stage-specific biases. A smaller learning rate leads to more conservative updates, resulting in a smoother training process, but the characterization of key anomaly patterns may be diluted, leading to less stable improvement in discrimination. Therefore, the multiple inflections in the curve are not unexpected; they reflect the dynamic balance between the multimorphic nature of link anomaly patterns and the optimization process.

It is noteworthy that the curve reaches a higher level at smaller learning rates, indicating that stable, incremental optimization is more conducive to forming reliable discrimination boundaries in this task. This aligns with the balance emphasized in the previous overall comparative experiments; that is, the method does not rely on aggressive updates for short-term gains, but rather gradually solidifies

multi-scale information into the representation through a more robust training process, thereby achieving a better trade-off between covering anomalies and suppressing false positives. In summary, this sensitivity result also suggests that when migrating under different data distributions or acquisition conditions, fine-grained searches should be prioritized in regions with smaller learning rates to reduce the risk of significant performance fluctuations.

Noise disturbance is one of the most common non-ideal factors in telemetry data, potentially originating from sampling jitter, measurement errors, transient congestion, or reporting delays. For anomaly detection models based on multi-scale time-series representations, noise alters the local fluctuation morphology and feature distribution, thus affecting the model's ability to distinguish between anomalous patterns and normal fluctuations. To evaluate the robustness and usability of the method under different disturbance intensities, it is necessary to systematically examine the impact of noise level changes on detection performance.

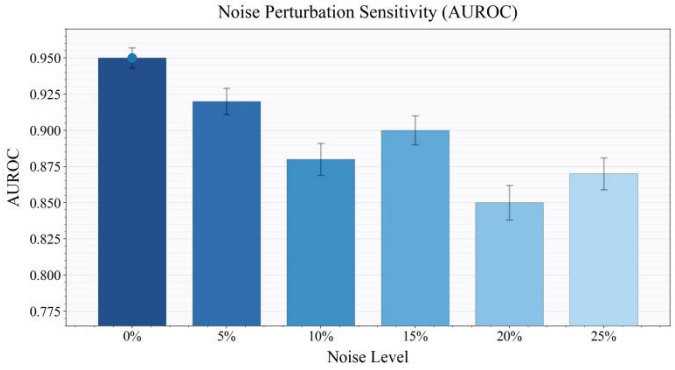

Figure 4. Data sensitivity experiment of noise disturbance intensity to AUROC

As can be seen from Figure 4, the model's discrimination ability generally declines with increasing noise perturbation, but there is a brief rebound in the medium perturbation range. This indicates that the method does not simply rely on the absolute accuracy of the input, but has a certain degree of noise tolerance and feature redundancy utilization. When the perturbation continues to increase, the performance drops significantly again, reflecting that high-intensity noise can destroy the temporal consistency and cross-scale alignment of the link features, further blurring the boundary between normal fluctuations and abnormal patterns. Combining this with the previous discussion on learning rate sensitivity, it seems that data-side uncertainty amplifies the optimization difficulty and representation instability. The combined effect of these two factors increases the risk of misjudgment. Therefore, in practical deployment, it is necessary to control the noise sources in the acquisition link or introduce more robust denoising and calibration strategies to maintain alarm quality.

6. Limitation

While this study presents a unified framework for multi-scale temporal modeling and backend anomaly detection of microservice links, certain boundary conditions remain. First, the method primarily relies on distributed tracing and aggregated link features to characterize system states. When there are persistent sampling biases on the monitoring side, missing tracing points for critical services, or links being interrupted by asynchronous messages and cross-domain calls, the integrity and alignment of the input sequence decrease, thus affecting the stable representation of anomaly patterns. Furthermore, in multi-tenant or frequently iterating production environments, service topologies and call paths may change rapidly. The dependency structure learned by the model during training may not reflect the latest state promptly, resulting in insufficient agility in responding to new anomalies or structural mutations.

Second, the framework emphasizes multi-scale fusion and global dependency modeling, which improves the coverage of complex anomalies but also introduces certain computational and engineering burdens. Especially in high-throughput scenarios, more refined window partitioning, sampling strategies, and online inference scheduling are needed to ensure controllable latency overhead. Meanwhile, current anomaly scoring focuses on the reliability and stability of detection, but its ability to pinpoint and explain root causes still relies on additional operational knowledge or external analysis tools, making it difficult to directly provide actionable causal chains and remediation suggestions. Therefore, how to further enhance online adaptability and interpretable diagnostic capabilities without significantly increasing costs remains a direction worthy of continued exploration.

7. Conclusion

This framework aims to provide more stable and usable anomaly identification capabilities under conditions of complex call dependencies, strong noise fluctuations, and multi-source heterogeneous features. By incorporating short-term local fluctuations and long-term global background into a unified representation space, the framework can extract key risk signals from the temporal structure of inter-service propagation, thereby mitigating the problem of single-scale modeling easily failing under link jitter, sudden congestion, and structural changes. Overall, this work provides a data organization method and modeling paradigm for link-level anomaly detection that is closer to engineering practice, laying a methodological foundation for subsequent online monitoring, automated operation, and maintenance.

From an application value perspective, microservice architecture has become the mainstream form of cloud-native architecture. Link anomalies often exhibit characteristics such as cross-service propagation, fragmented representation, and great difficulty in localization, directly impacting business continuity and user experience. The multi-scale fusion and global dependency modeling emphasized in this framework enable anomaly detection to move beyond single-service indicator thresholds or local segment discrimination, allowing for a more consistent characterization and prioritization of risks based on the overall behavior of the end-to-end call chain. These capabilities are directly significant for stability engineering, capacity planning, fault early warning, and change management. They help reduce the troubleshooting costs associated with false alarms, improve alarm reliability and response efficiency, and provide reliable foundational signals

for building a more automated cloud operations and maintenance (O&M) system.

Meanwhile, this paper further illustrates the potential role of link data in intelligent O&M. Distributed tracing is not only used for post-incident troubleshooting but can also serve as a core data asset for real-time monitoring and risk assessment. A link-centric representation approach more naturally carries system attributes such as service dependencies, critical paths, and propagation effects, making the detection model more aligned with real-world fault mechanisms. For backend systems with complex dependencies and frequent updates, this ability to understand system status at the link level can drive a shift from passive troubleshooting to proactive early warning and pre-emptive governance, thereby improving platform-level reliability and business delivery quality. It has transferable and widespread value in large-scale cloud services, internet businesses, and enterprise-level microservice platforms.

Looking to the future, there are still many directions worth exploring further. Firstly, it enhances online adaptation and continuous learning capabilities, enabling models to respond more quickly to service topology changes, version iterations, and load pattern shifts, reducing reliance on offline retraining and improving stability during cross-environment migration. Secondly, it further integrates richer operational semantic information, such as logs, metrics, and event streams, forming a unified diagnostic perspective with link representation, thereby driving the shift from anomaly detection to more actionable root cause analysis and remediation recommendations. Thirdly, it optimizes inference overhead and scheduling strategies around engineering implementation, ensuring the framework maintains availability and alignment accuracy under high throughput and low latency constraints. As the cloud-native ecosystem continues to evolve, structured time-series modeling oriented towards links is expected to become a crucial technological pillar of intelligent operations and maintenance, providing continuous support for more reliable and self-healing backend systems.

References

[1] Shi K, Li J, Liu Y, et al. BSDG: Anomaly Detection of Microservice Trace Based on Dual Graph Convolutional Neural Network[C]//International Conference on Service-Oriented Computing. Cham: Springer Nature Switzerland, 2022: 171-185.

[2] Li Y, Lu Y, Wang J, et al. TADL: Fault localization with transformer-based anomaly detection for dynamic microservice systems[C]//2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER). IEEE, 2023: 718-722.

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

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

[5] Y. Ni, "Learning Multi-Scale Generative Representations for Cloud Performance Anomaly Detection via Self-Distillation", 2024.

[6] Z. Wang, "Federated Multi-Scale Representation Learning for Privacy-Aware Log Anomaly Detection in Distributed Cloud Environments", 2024.

[7] C. Wang, "Towards Intelligent Backend Operations via Unified Temporal Representation and Prediction Modeling", Artificial Intelligence and Computing Innovations, vol. 4, no. 3, 2024.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] Zhang S, Pan Z, Liu H, et al. Efficient and robust trace anomaly detection for large-scale microservice systems[C]//2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE). IEEE, 2023: 69-79.

[13] 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.

[14] 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.