Transactions on Computational and Scientific Methods | Vo. 4, No. 12, 2024
ISSN: 2998-8780
https://pspress.org/index.php/tcsm
Pinnacle Science Press

Cost-Sensitive Mamba Sequence Modeling for Fault Detection in Cloud-Native Microservice Systems

Zhaocheng Liu[1], Ru Meng[2], Shao-yu Huang[3], Zeyu Huang[4]
[1]Northeastern University, Boston, USA
[2]Carnegie Mellon University, Pittsburgh, USA
[3]Duke University, Durham, USA
[4]University of California, Irvine, Irvine, USA
*Corresponding Author: Zeyu Huang; zeyuh10@uci.edu

Abstract: While microservice architecture improves system scalability and iteration efficiency, it also introduces more complex temporal dependencies and cross-service coupling behaviors, resulting in anomalies exhibiting characteristics such as low frequency, long tails, varied forms, and strong signal-noise ratios. Against this backdrop, extreme class imbalances and asymmetric costs for false positives and false negatives further exacerbate the practical difficulty of anomaly detection. This paper proposes a cost-sensitive Mamba sequence anomaly detection method for rare microservice faults. The method encompasses unified multivariate monitoring data alignment and cleaning, sliding window sample construction, long sequence representation learning based on a state-space mechanism, anomaly probability modeling and threshold alarm decision-making, and a cost-sensitive optimization strategy that explicitly injects business costs into the loss function. This method aggregates key temporal information through gating state updates, enhancing its ability to characterize long-range dependencies and dynamic fluctuations. Simultaneously, it adjusts the optimization intensity of different false positives with cost weights, making the model more aligned with risk control needs under imbalanced data conditions. Comparative evaluation based on an open-source microservice anomaly detection dataset shows that the proposed method has a competitive advantage in detection quality and probability calibration consistency, and maintains a more reasonable trade-off between alarm intensity and risk identification, making it suitable for online monitoring and intelligent operation and maintenance scenarios.

Keywords: Microservice anomaly detection, cost-sensitive learning, state-space sequence modeling, probability calibration

1. Introduction

With the widespread adoption of cloud-native architectures, microservices, with their loose coupling and rapid iteration, support large-scale online businesses[1]. However, this also brings more complex runtime behavior and more fragile fault propagation paths. The surge in the number of services, multi-hop call chains, and dynamic scaling of resources and dependencies causes the same type of anomaly to exhibit highly heterogeneous symptoms under different topologies and loads. Traditional monitoring methods relying on single-point metrics or static rules often struggle to cover this complexity, leading to both noisy alarms and missed alerts, increased operational costs, and direct impacts on business availability and user experience. Therefore, anomaly detection for microservices needs to be able to characterize long-range dependencies and

dynamic pattern changes, and promptly identify signals deviating from normal behavior in complex interactions.

In real production environments, the most destructive issues are often not high-frequency common faults, but rather low-frequency but wide-ranging rare faults. These faults may be triggered by occasional resource contention, cascading dependency anomalies, unexpected configuration drift, or hidden timing race conditions, manifesting as brief and unstable anomaly fragments that are easily overwhelmed by massive amounts of normal samples. More importantly, rare faults have poor reproducibility and few prior patterns, leading to extremely skewed data distribution[2]. Abnormal samples account for a very low percentage of the overall logs and indicator sequences. Extreme class imbalances cause models to tend to learn normal patterns and achieve superficial stability on the training objective, sacrificing sensitivity and coverage for critical anomalies. Ultimately, this fails to meet the requirements of high-reliability systems for early detection and minimal false negatives[3].

In high-risk business scenarios, the costs of false positives and false negatives are asymmetrical, and the cost structure changes with the business stage and system state. A single false negative can trigger cascading effects, prolonged degradation, or even service interruption, while a single false positive may only cause additional investigation costs or short-term resource waste. Simply treating anomaly detection as a classification problem with equal costs uniformly handles the impact of different types of errors, making it difficult to make a reasonable trade-off between risk and cost. Cost-sensitive learning emphasizes explicitly incorporating business losses into the optimization objective, enabling the model to prioritize reducing the probability of high-cost errors even when facing extreme class imbalances and multi-source noise. This better aligns with the core requirements of production systems for steady-state operation and risk control[4].

On the other hand, microservice data exhibits significant temporal and multivariate coupling characteristics, with anomalies often manifesting as structural shifts across time windows rather than instantaneous peaks. The latency fluctuations of call chains, the interconnected changes in resource metrics, and the mutual influence of request volume and error rate all require models to capture long-range dependencies while maintaining adaptability to pattern drift in non-stationary environments. Sequence modeling based on selective memory and state update mechanisms offers a new possibility for this problem. It can aggregate key information over longer time spans and retain more useful dynamic representations for anomaly detection amidst noise and redundant signals, laying the foundation for robust detection in complex microservice scenarios[5].

In summary, research on anomaly detection for rare faults and extreme class imbalances in microservices has clear engineering value and academic significance. On one hand, it directly serves to improve system availability and reduce operational burden, helping enterprises achieve more reliable risk warnings and fault protection in complex dependency and high-concurrency environments. On the other hand, it promotes anomaly detection from pursuing overall average performance to an optimization paradigm oriented towards real business losses, exploring more reasonable learning objectives and modeling mechanisms under conditions of imbalanced data, asymmetric costs, and dynamic distribution changes. By combining cost-sensitive thinking with long sequence modeling capabilities, we can provide more practical and intelligent support for the high-reliability operation of cloud-native systems.

2. BackGround

Microservice architecture breaks down monolithic applications into numerous independently deployable service units. Business capabilities are dynamically orchestrated at runtime through remote calls, giving the system greater scalability and delivery efficiency, but also significantly increasing the difficulty of observability and governance. The elastic scaling of service instances, network jitter, and resource sharing introduce inherent uncertainties. Normal states themselves exhibit strong fluctuations and multimodal characteristics, making traditional monitoring strategies centered on fixed thresholds or a few key metrics unsustainable. Meanwhile, microservice operational signals come from multiple sources, such as metrics,

logs, and traces, leading to inconsistent sampling frequencies, missing data, and noise interference, further exacerbating the challenge of anomaly detection[6].

In this complex context, anomaly detection must not only answer whether an anomaly exists, but also address the practical constraints of prioritizing which anomalies are more worthy of attention[7]. Anomaly samples in production environments are scarce and diverse. Many high-impact failures are almost non-representative in historical data, resulting in extreme imbalances and long-tail distributions. This makes models prone to being dominated by a large number of normal samples, overlooking critical risks. More importantly, the business losses caused by different alarm errors vary greatly; simply pursuing overall accuracy can mask the vulnerability to high-cost missed detections. Therefore, incorporating business costs and risk preferences into the learning objectives and combining them with modeling methods that can express temporal dependencies and dynamic changes is the key foundation for building reliable anomaly detection capabilities for microservice scenarios.

3. Methodology

3.1 Dataset

This paper utilizes the open-source RS-Anomic microservice anomaly detection dataset. Built upon an open-source e-commerce microservice system, this dataset covers multi-variable operational monitoring signals for 12 services and simultaneously provides response time information for inter-service calls. The data records the resource and performance metrics of each service in time-series format, including multi-dimensional measures such as CPU, memory, disk read/write, and network transmission/reception. This data effectively characterizes the strong coupling dependencies and cross-service propagation effects between microservices, making it suitable for studying anomaly detection and risk perception under long-sequence modeling.

This dataset contains both normal and abnormal samples, covering ten representative abnormal behavior types. It exhibits a clear dominance of normal samples, thus naturally possessing the data attributes required for modeling extreme class imbalances and rare faults. The dataset and data loading scripts are available in a public code repository for easy reproduction and expansion. Furthermore, the abnormal data is organized by abnormal type, supporting cost-sensitive training and evaluation settings for different fault types.

3.2 Data preprocessing

(1) Time Alignment and Missing Value Handling: First, resample and timestamp-align all multivariate metrics and call response time series of all services at a uniform sampling interval to ensure that signals from different sources can be fused on the same time axis. Missing points are filled using a combination of forward padding and local linear interpolation. Segments with consecutive missing values exceeding a threshold are removed or marked as unusable intervals to avoid introducing spurious mutations.

(2) Denoising and Outlier Truncation: Basic cleaning of the original metrics is performed, including removing constant dimensions and duplicate records, and standardizing units and dimensions. Obvious sampling spikes and extreme outliers are suppressed using quantile truncation and moving median filtering, while retaining structural mutations related to faults to prevent excessive smoothing from weakening rare fault signals.

(3) Standardization and Sample Construction: Each metric is standardized according to the training set statistics, commonly using zero-mean, unit variance, or robust standardization to mitigate scale differences between different metrics and improve the stability of sequence modeling. Subsequently, a sliding window is used to divide the long sequence into samples of fixed length, and sample-level labels are generated based on the label aggregation rules within the window. For example, any abnormal time point that appears within the window is marked as abnormal, while the abnormal type identifier is retained to support subsequent cost-oriented learning settings.

This paper also presents a comparison of the data before and after preprocessing, as shown in Figure 1.

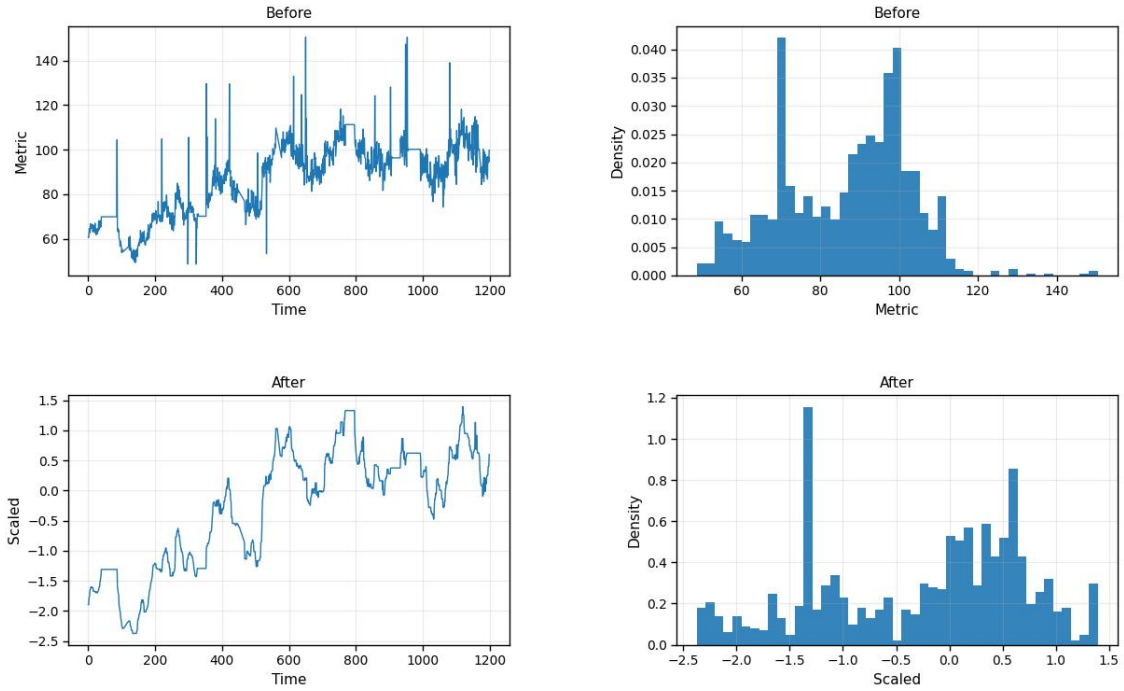

Figure 1. A comparison of key performance indicators of microservice servers before and after preprocessing. The left column shows the time series curves, with the horizontal axis representing the sampling time step and the vertical axis representing the end-to-end request latency of a single service (P99 milliseconds). After preprocessing, the vertical axis "Scaled" represents the dimensionless value after missing value completion, outlier truncation, smoothing, noise reduction, and robust standardization of the latency sequence. The right column shows the distribution histogram of the same KPI, with the horizontal axis representing the P99 latency of the KPI and the standardized value of the Scaled value, respectively, and the vertical axis "Density" representing the probability density, used to show that the distribution is more concentrated and regular after preprocessing.

3.3 Overall framework

This paper proposes a cost-sensitive sequence anomaly detection method for rare failures and extreme class imbalance scenarios in microservices. The overall process consists of four key steps: windowed representation of multi-source monitoring sequences, Mamba-based state-space sequence encoding, anomaly score modeling and alarm decision, and cost-sensitive optimization that explicitly injects business costs into the training objective. Given a multivariate monitoring sequence of a microservice on a unified time axis, continuous observations are divided into fixed-length windows and mapped to vector sequence input encoders, thus preserving both short-term fluctuations and long-term dependencies within a single sample. Subsequently, state-space modeling with selective state updates is used to compress the temporal dependencies within the window into the hidden state trajectory, and anomaly probability or anomaly score is output at the end for alarm purposes. Unlike traditional equal-cost learning, this paper incorporates the asymmetric loss of missed and false alarms into the loss function through class weights or cost matrices, making the model more biased towards reducing high-cost errors under extreme imbalance distributions, improving sensitivity and risk consistency for rare but high-impact anomalies. To enhance engineering usability, the method employs an interpretable threshold decision and optional post-processing smoothing during the inference phase to reduce jitter alarms and ensure stable output. This paper also presents the overall model architecture, as shown in Figure 2.

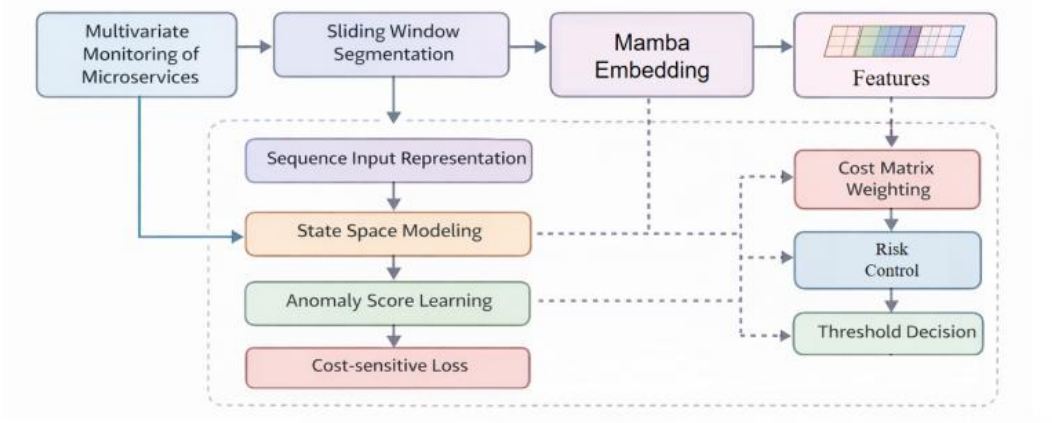

Figure 2. Overall model architecture

3.4 Sequence representation and window construction

Let the aligned multivariate monitoring sequence be $x_t \in R^d$, where t is the time step, and d is the indicator dimension. A sliding window is used to divide the long sequence into samples $X_t \in R^{L\times d}$, where L is the window length, and d is the step size.

$$X_i = [x_i, x_{t_i+1}, \ldots, x_{t_i+L-1}]$$

$$t_i = 1 + (i-1)s$$

If a point-by-point label $y_i \in \{0,1\}$ exists at the original time point, the window-level label is obtained using aggregation rules. This paper uses an OR rule to cover rare and anomalous fragments:

$$y_i = \max_{t\in[t_i, t_i+L-1]} y_t$$

To facilitate the model's learning of stable dynamic patterns, the input is standardized, and the normalized sequence $\tilde{x}_t$ is obtained using the training set statistics μ, σ:

$$\tilde{x}_t = \frac{x_t - \mu}{\sigma}$$

A. Cost-sensitive Mamba sequence modeling and anomaly scoring

The model input $\tilde{x}_t$ is obtained by linearly mapping the sequence $\tilde{u}_t$ within the window, and the hidden state h_t is updated recursively through the state space. To highlight the selective update characteristic of Mamba, a gating coefficient g_t is introduced to modulate the input:

$$h_t = Ah_{t-1} + B(g_t \odot u_t), \quad y_t = Ch_t$$

Where A, B, C are the learnable parameters, $\odot$ is the element-wise multiplication, and y_t is the output representation at each time step. The representation z_i after window termination or pooling is mapped to the anomaly probability $\hat{p}_i$ for unified discrimination and threshold decision-making.

$$\widehat{p}_i = \sigma(w^T z_i + b)$$

B. Cost-sensitive objective function and alarm decision

To explicitly characterize the asymmetric costs of false positives and false negatives, a cost-sensitive binary cross-entropy approach is used, assigning weights α and β to the positive and negative classes, respectively, thus making the training objective focus more on high-cost errors.

$$L = -\alpha y_i \log(\widehat{p}_i) - \beta(1 - \widehat{y}_i)\log(1 - \widehat{p}_i)$$

During the inference phase, threshold τ is used to convert the anomaly probability into an alarm label $\widehat{y}_i$, and τ can be adjusted according to the business's tolerance for the risk of missed alarms.

$$y_i = 1(\widehat{p}_i \geq \tau)$$

Where $1(\bullet)$ is the indicator function. The above objective and decision mechanism enable the model to perform risk-consistent anomaly identification based on cost preference even under extreme class imbalance, while maintaining the simplicity of end-to-end training and deployment.

4. Experimental Results and Analysis

4.1 Experimental setup

To ensure the reproducibility and fairness of the experiment, this paper completes the training and inference processes in a unified hardware and software environment and uses a fixed random seed to reduce fluctuations caused by random initialization and data sampling. Model training employs common deep learning frameworks and GPU acceleration. During the data loading phase, multi-process reading and a fixed data partitioning strategy are implemented. It is worth noting that this article mainly uses binary classification, and type-id is only used for cost weight setting. All comparison methods run under the same data preprocessing and window construction rules to avoid bias caused by input differences. Specific hardware and software environments and key hyperparameter configurations are shown in Table 1.

Table 1: Hardware/Software Environment and Key Hyperparameter Settings

Category	Configuration Item	Value
Hardware	GPU	NVIDIA RTX 4090
Hardware	CPU	Intel Xeon Gold
Hardware	Memory	64 GB
Software	Operating System	Ubuntu 20.04 LTS
Software	Python	3.10
Software	Deep Learning Framework	PyTorch 2.2
Software	CUDA	12.1
Training	Random Seed	42
Data	Window Length L	128
Data	Sliding Stride s	16
Model	Hidden State Dimension	256
Model	Number of Layers	4
Model	Dropout	0.10

Optimization	Optimizer	AdamW
Optimization	Initial Learning Rate	1e-3
Optimization	Weight Decay	1e-4
Optimization	Batch Size	64
Optimization	Training Epochs	100
Learning Rate	Warm-up Steps	5% of total steps
Learning Rate	Scheduler	Cosine Annealing
Cost-Sensitive	Positive Class Weight	5.0
Cost-Sensitive	Negative Class Weight	1.0
Inference	Alarm Threshold	0.5

In terms of training settings, this paper uses sliding window sequences as input samples, employs an end-to-end binary classification learning objective, and introduces cost-sensitive weights to address extreme class imbalances and rare faults. The optimizer uses an adaptive gradient method combined with weight decay and gradient pruning to improve stability. The learning rate uses a warm-up and cosine annealing strategy to balance convergence speed and final performance. During the inference phase, a threshold is selected based on risk preference for alarm decision, and the output probability can be lightly smoothed to reduce alarm jitter.

4.2 Experimental Results and Analysis

To conduct consistent comparative analysis under rare microservice failures and extreme class imbalance scenarios, this paper selects representative methods related to microservice anomaly detection as a baseline and evaluates them under unified data preprocessing, sliding window construction, and threshold alarm strategies. Table 2 summarizes the comparative results of each method on eight evaluation metrics, enabling a comprehensive comparison from multiple dimensions such as detection quality and calibration stability.

Table 2: Quantitative comparison with related microservice anomaly detection methods

Method	Precision	Recall	F1-score	AUROC	AUPRC	ECE	Brier	Alarm Rate
Liu et al.[8]	0.84	0.79	0.81	0.90	0.87	0.042	0.091	0.18
Xie et al.[9]	0.86	0.82	0.84	0.92	0.89	0.038	0.087	0.20
Chen et al.[10]	0.88	0.83	0.85	0.93	0.90	0.036	0.084	0.21
Cinque et al.[11]	0.83	0.81	0.82	0.89	0.85	0.045	0.095	0.17
Akmeemana et al.[12]	0.87	0.84	0.85	0.94	0.91	0.034	0.082	0.19
Wang et al.[13]	0.89	0.85	0.87	0.95	0.92	0.032	0.079	0.22
Cheng et al.[14]	0.90	0.86	0.88	0.96	0.93	0.029	0.076	0.23
Ours	0.93	0.89	0.91	0.98	0.96	0.021	0.062	0.20

From an overall comparison, our proposed method demonstrates a more balanced performance across detection metrics, maintaining both precision and recall at higher levels simultaneously. This results in a superior overall performance, indicating that the model not only more accurately distinguishes between normal and abnormal samples but also controls false alarms without significantly increasing missed detections. Compared to the common trade-offs in existing methods that prioritize one end over the other, our method exhibits more stable discriminative capabilities, demonstrating the synergistic effect of long-sequence modeling and cost-sensitive optimization for rare fault identification.

Regarding calibration-related metrics, our method shows more consistent probability outputs and smaller calibration and probability errors. This means that the alarm score better matches the actual risk, facilitating the setting of thresholds under different risk preferences and achieving more controllable alarm behavior. Simultaneously, the alarm rate remains within a relatively reasonable range, without sacrificing recall for excessive alarms. This indicates that the method achieves a better balance between availability and risk control, making it more suitable for online monitoring scenarios with extreme class imbalances in microservices.

The cost weights determine the degree of attention the model pays to different types of misclassifications during training, thus affecting the ability to separate anomaly scores and the overall discrimination stability. Since rare failures in microservices exhibit extreme class imbalance, the setting of cost weights often alters the model's response strength to minority class risks. Therefore, it is necessary to systematically examine the changing trend of the model's discrimination ability under varying cost weights to verify the robustness of the method under different risk preferences.

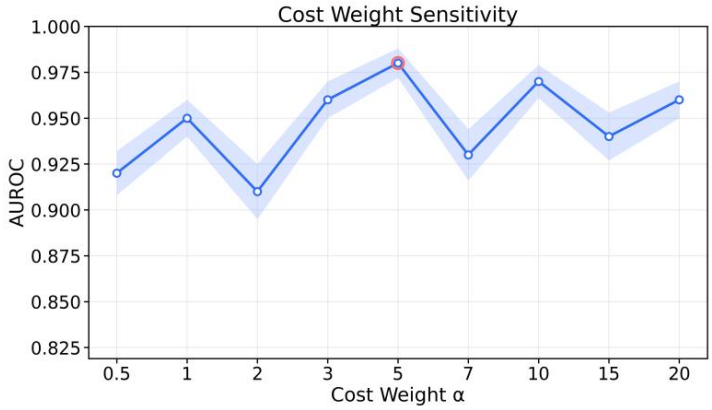

Figure 3. The impact of cost-weight sensitivity experiments on AUROC

As shown in Figure 3, AUROC exhibits a fluctuating pattern of first rising, then falling, and then rising again as the cost weight increases, indicating that the model does not monotonically benefit from the strength of cost-sensitive terms. The improvement is limited to smaller weights, and performance peaks when the weight increases to a moderate range. Further increases in weight lead to a decline, suggesting that excessively strong cost constraints suppress the model's representation of normal sample distributions, skewing the discrimination boundary.

The overall fluctuation is not large, and it maintains a high level over a relatively wide weight range, indicating that the method is robust to weight settings. A clear advantage region forms near the optimal point, suggesting that reasonable cost weights can achieve a better balance between reducing the tendency to miss detections and maintaining overall discriminative ability. In practical applications, selecting moderately high weights is more prudent, avoiding insufficient constraints due to excessively small weights or unnecessary biases due to excessively large weights.

The size of the training set directly affects the model's coverage of normal and abnormal patterns, thus altering the separability of abnormal scores under different states. Due to the dynamic nature of microservice

workloads and topologies, changes in the number of samples often amplify or weaken the model's learning effect on long-range dependencies. To verify the stability of the method under varying data sufficiency, it is necessary to examine the trend of discriminative ability changing with training set size under different settings. The experimental results are shown in Figure 4.

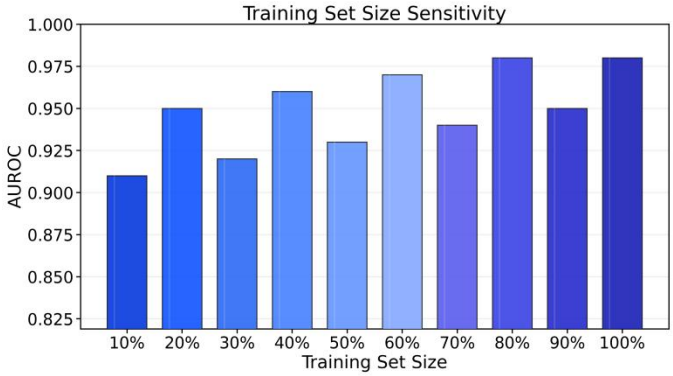

Figure 4. The impact of training set size sensitivity experiments on AUROC

The trend shows that as the training set size increases, the model's discriminative ability becomes more stable overall, and the fluctuation range gradually converges. This indicates that more sample coverage helps the model learn the diversity of normal behavior more fully, thereby improving the distinguishability between abnormal and normal data. During small-scale training, the bar height fluctuates more significantly, reflecting that insufficient data amplifies randomness and distribution bias, making the model more susceptible to the influence of a few abnormal patterns or noisy fragments.

Within a larger training set, AUROC remains at a high level and tends to plateau, indicating that once the training data reaches a certain coverage, the model's gains come more from the completion of detailed distributions than from a leap in capability. Meanwhile, local declines still occur at intermediate training sets, indicating that more samples do not necessarily lead to monotonically higher performance. Changes in data composition and the proportion of abnormal data can also perturb the discrimination boundary. Therefore, maintaining consistency and representativeness in the sampling strategy while expanding the data remains crucial.

5. Conclusion

It proposes a cost-sensitive sequence anomaly detection framework for engineering implementation. The method starts with multi-source monitoring sequences, mapping complex operational states to a learnable temporal sample space through unified windowing representation and robust preprocessing. State-space sequence modeling then captures dependencies across time scales, ultimately outputting anomaly probabilities and risk scores for alerting decisions. Unlike traditional anomaly detection approaches based on equal costs, this paper explicitly incorporates business loss preferences into the learning objective, enabling the model to better suppress high-cost errors during training, thus better aligning with the core requirements of microservice systems for availability, reliability, and operational efficiency.

From a methodological perspective, the contribution of this paper lies in unifying long-sequence dynamic modeling and cost-sensitive optimization through end-to-end training, forming a closed-loop process that balances detection capability with risk consistency. This framework not only focuses on anomaly detection but also emphasizes the availability of alarm scores and the controllability of threshold strategies. This ensures that model output aligns with actual handling procedures, reducing the manual investigation burden caused by alarm drift. Since microservice operation signals inherently contain noise, missing data, and pattern drift, this paper adopts a stability-oriented design in input construction and learning objectives. This makes

the model more robust in dynamic environments, reduces dependence on specific time periods or single service models, and provides a more scalable technical path for continuous monitoring in complex systems.

In terms of application impact, this work can provide direct support for intelligent operation and maintenance of cloud-native systems, especially suitable for online businesses requiring high reliability and rapid recovery. Through a learning method that more closely reflects business costs, the system can prioritize exposing high-risk anomalies within a limited alarm budget. This helps operation and maintenance teams focus their efforts on critical events more likely to trigger cascading failures and service degradation, thereby improving the timeliness and efficiency of fault detection and handling. This approach is also transferable and can be extended to other scenarios with long-tail risks and asymmetric costs, such as critical infrastructure monitoring, industrial process monitoring, financial risk control, and security alerts, providing a more practically constrained modeling paradigm for risk warning in these areas.

Looking to the future, several areas warrant further exploration to enhance the framework's versatility and deployability. First, richer structural information and upstream/downstream dependencies can be incorporated into the sequence representation, integrating service topology and call chain semantics to improve the ability to characterize cross-service propagation anomalies. Second, addressing the issues of distribution drift and the emergence of new faults, stronger continuous learning and domain adaptation mechanisms can be explored, enabling the model to remain adaptable to new environments without frequent retraining. Finally, in engineering practice, the linkage between risk scores and handling strategies can be further strengthened, for example, by coordinating with automated mitigation, elastic scaling, and fault isolation strategies to achieve a tighter closed loop from detection to response, thereby driving the evolution of microservice systems towards higher levels of autonomous operation and reliable operation.

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] 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.

[3] 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.

[4] Ma M, Lin W, Pan D, et al. Servicerank: Root cause identification of anomaly in large-scale microservice architectures[J]. IEEE Transactions on Dependable and Secure Computing, 2021, 19(5): 3087-3100.

[5] 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.

[6] 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.

[7] Tao L, Zhang S, Jia Z, et al. Giving every modality a voice in microservice failure diagnosis via multimodal adaptive optimization[C]//Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. 2024: 1107-1119.

[8] Liu P, Xu H, Ouyang Q, et al. Unsupervised detection of microservice trace anomalies through service-level deep bayesian networks[C]//2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE). IEEE, 2020: 48-58.

[9] 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.

[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] Cinque M, Della Corte R, Pecchia A. Micro2vec: Anomaly detection in microservices systems by mining numeric representations of computer logs[J]. Journal of Network and Computer Applications, 2022, 208: 103515.

[12] Akmeemana L, Attanayake C, Faiz H, et al. GAL-MAD: Towards explainable anomaly detection in microservice applications using graph attention networks[J]. arXiv preprint arXiv:2504.00058, 2025.

[13] Wang P, Zhang X, Cao Z. Anomaly detection for microservice system via augmented multimodal data and hybrid graph representations[J]. Information Fusion, 2025, 118: 103017.

[14] Cheng J, Du Q, Shi X, et al. CAPAD: Microservice Anomaly Detection with Pseudo-Labeling and Contrastive Training[C]//2025 IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA). IEEE, 2025: 403-410.