Article

Not peer-reviewed version

AI-Driven Performance Degradation Identification via Self-Supervised Spatiotemporal Graph Modeling in Microservice Systems

Yuchen Liu *

Posted Date: 6 February 2026

doi: 10.20944/preprints202602.0480.v1

Keywords: self-supervised learning; performance degradation detection; microservice architecture; spatiotemporal feature modeling

 doi:10.20944/preprints202602.0480.v1

Article

AI-Driven Performance Degradation Identification via Self-Supervised Spatiotemporal Graph Modeling in Microservice Systems

Yuchen Liu

University of Pennsylvania, Philadelphia, USA; yuchenliu893@gmail.com

Abstract

This paper proposes a self-supervised performance degradation identification model to address the challenges of high-dimensional heterogeneous data, complex dependency structures, and dynamic non-stationarity in large-scale microservice architectures. The model takes multi-source monitoring data as input and first performs semantic alignment among different metrics through multidimensional feature embedding and projection layers. An adaptive dynamic graph convolutional network is then employed to capture the topological dependencies and interaction features among service nodes, constructing time-varying structural representations. In the temporal modeling stage, a gated recurrent unit-based embedding mechanism is introduced to jointly characterize long-term dependencies and local fluctuations of performance evolution, while a residual fusion structure enhances the stability of feature propagation. To improve feature discrimination under unsupervised conditions, the model adopts a contrastive learning optimization strategy and utilizes a temperature adjustment mechanism to strengthen the distinction between positive and negative samples in the latent space, enabling adaptive aggregation and recognition of degradation patterns. Furthermore, multiple hyperparameter sensitivity experiments are conducted to systematically evaluate the effects of learning rate, residual coefficient, temperature parameter, and monitoring sampling interval on model performance. Experimental results show that the proposed model outperforms mainstream methods in accuracy, precision, recall, and F1-score, achieving efficient and stable identification of performance degradation in complex microservice systems under unsupervised settings, thus providing a practical solution for intelligent operations and maintenance.

Keywords: self-supervised learning; performance degradation detection; microservice architecture; spatiotemporal feature modeling

I. Introduction

With the rapid development of cloud computing and containerization technologies, microservice architecture has become the mainstream form of modern distributed systems. Compared with traditional monolithic applications, microservice systems emphasize service autonomy, modular decoupling, and elastic scalability. Through fine-grained service partitioning and lightweight communication mechanisms, they provide high concurrency and high availability for complex business processes. However, as system scale expands and the number of services grows exponentially, the operating environment of microservices becomes highly dynamic and complex. Frequent changes in service dependencies, deep coupling in invocation chains, and distributed scheduling of heterogeneous resources create multidimensional interference and uncertainty in system performance [1]. When the system runs at a large scale, minor latency fluctuations or resource bottlenecks can be amplified along the dependency chain, leading to system-level performance degradation. Such degradation often manifests as increased response time, reduced throughput, or rising error rates, which not only affect user experience but also cause significant economic losses and

 doi:10.20944/preprints202602.0480.v1

resource waste. Therefore, accurately identifying and providing early warnings of performance degradation in complex microservice architectures has become a critical research direction in intelligent operations and maintenance [2].

Traditional performance monitoring and anomaly detection methods largely rely on expert-defined rules or supervised learning for classification. Although these methods achieve acceptable results in small-scale systems or specific scenarios, they face major limitations in large-scale microservice environments. Rule-based approaches cannot adapt to the frequent evolution of service topologies and struggle with cross-component dependencies and dynamic load variations. Supervised learning methods depend on large quantities of labeled data, yet performance degradation events in real systems are difficult to label precisely, leading to data scarcity. Moreover, monitoring data from logs, traces, and metrics is typically high-dimensional, heterogeneous, and time-varying. As a result, conventional models fail to learn stable and generalizable representations under noisy conditions. Hence, building an unsupervised and adaptive model that can learn service dependencies and performance evolution patterns is key to addressing large-scale performance degradation problems.

In recent years, the rise of artificial intelligence and deep learning in system management has provided new opportunities for performance degradation identification. Self-supervised learning has emerged as an effective paradigm that enables models to learn robust representations without manual annotations by constructing pseudo-labels or contrastive objectives. This approach is particularly suitable for microservice environments, where normal samples are abundant but degraded samples are rare. It can capture latent performance change patterns from large volumes of monitoring data. In addition, microservice systems exhibit strong spatial and temporal dependencies. On one hand, service nodes are connected by complex topological relationships, and performance fluctuations can propagate along invocation chains. On the other hand, performance metrics evolve dynamically over time, combining short-term fluctuations and long-term trends. Therefore, integrating self-supervised learning with spatiotemporal modeling allows structured perception of performance degradation from a global perspective, providing high-quality representations for root cause analysis and intelligent optimization. Compared with traditional methods, self-supervised approaches can maintain adaptability in evolving environments, significantly improving the timeliness and generalization of degradation recognition [4].

The complexity of large-scale microservice systems is reflected not only in the vast number of services but also in the explosive growth of data generation rates and feature dimensions. With the introduction of container orchestration and elastic scaling, system states become highly dynamic, and the frequent creation and destruction of service instances lead to data characterized by temporal variation, abrupt changes, and non-stationarity [5]. In this context, static modeling approaches cannot capture the dynamic evolution of performance patterns. Performance degradation is often a gradual process caused by the combined effects of resource contention, dependency chain congestion, and thread pool saturation, rather than a sudden anomaly. Therefore, realizing early identification and adaptive modeling of degradation trends under complex topologies and dynamic temporal conditions is essential for improving system reliability and user experience. Developing models with self-learning, self-evolving, and transferable capabilities can effectively address challenges posed by system scaling and workload fluctuations, serving as a cornerstone for intelligent operations and maintenance [6].

In summary, research on performance degradation identification for large-scale microservice architectures holds both theoretical and practical significance. Theoretically, it promotes the deep integration of self-supervised learning, spatiotemporal modeling, and distributed performance analysis, paving the way for intelligent systems with autonomous cognition. From an engineering perspective, it enables efficient and low-cost degradation identification from massive monitoring data, providing data-driven decision support for automated operations, capacity planning, and system optimization. Moreover, this research offers valuable insights for building reliable, interpretable, and scalable cloud-native infrastructures [7]. As system scale and business complexity continue to grow,

 doi:10.20944/preprints202602.0480.v1

performance degradation identification will evolve beyond a single anomaly detection task into a key component of full lifecycle system governance, exerting a profound impact on intelligent cloud management, edge collaboration, and multi-cloud resource orchestration.

II. Proposed Framework

The self-supervised performance degradation identification model proposed in this paper mainly consists of a multi-source feature representation module, a dynamic dependency modeling module, and a temporal embedding and contrastive optimization mechanism. First, the system extracts key indicators from the multi-dimensional monitoring data of microservices, including CPU utilization, memory usage, request latency, and error rate, and maps them uniformly to a multi-dimensional sequence representation $X \in R^{T \times d}$ with a time step of T and a feature dimension of d. To enhance the model's structural learnability, this paper employs a multi-head projection mechanism to transform the original features into a latent space representation:

$$H = MLP(XW_1 + b_1) \quad (1)$$

Where W_1 is the linear transformation matrix, b_1 is the bias term, and H represents the high-dimensional semantic features after nonlinear mapping. This step enables embedded modeling of monitoring metrics, providing a unified input space for subsequent topological dependency learning. Its overall model architecture is shown in Figure 1.

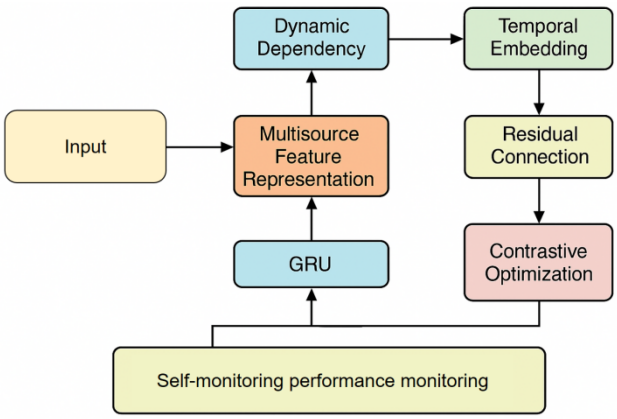


Figure 1. Overall model architecture diagram.

In service dependency modeling, the interaction of microservices can be represented as a dynamic directed graph $G_t \in (V, E_t)$, where V represents the set of service nodes and E_t represents the set of dependency edges at time step t. To capture the time-varying correlations between services, this paper introduces an adaptive graph convolution operator to aggregate node features:

$$Z_t = \sigma(\tilde{A}_t H_t W_g) \quad (2)$$

Where $\tilde{A}_t$ is the dynamically normalized adjacency matrix, W_g is the graph convolution weight matrix, and $\sigma(\cdot)$ is the activation function. By introducing an adaptive adjacency matrix generation mechanism, the model can dynamically update the dependency strength between services over time.

$$\tilde{A}_t = Softmax(RELU(H_t W_q H_t^T)) \quad (3)$$

This process enables adaptive dependency modeling at the structural level, which can flexibly depict the dynamic changes of the call chain at different runtime stages.

In the temporal feature modeling part, to better capture the temporal evolution features of performance degradation, the model adopts a position-encoded temporal embedding mechanism and realizes multi-scale temporal correlation modeling through a learnable temporal weight matrix:

$$S_t = GRU(Z_t, S_{t-1}) \quad (4)$$

 doi:10.20944/preprints202602.0480.v1

Here, S_t represents the hidden state at time step t, which can characterize the changing trends of performance metrics over short- and long-term time spans. To enhance sensitivity to time-series changes, a residual connection mechanism is further introduced:

$$F_t = S_t + LayearNorm(Z_t) \quad (5)$$

Through this design, the model can simultaneously capture local fluctuations and global trends over time, making the distribution of potential features more dynamically discernible.

In terms of optimization mechanism, this paper adopts a self-supervised contrastive learning strategy to maximize the feature consistency between similar samples and minimize the feature distance between dissimilar samples. Specifically, for any two time slices i, j, positive sample pairs are defined as feature representations (F_t, F_j) within adjacent time windows, and negative sample pairs are combinations of non-adjacent time windows. The self-supervised objective function is defined as:

$$L_{contrast} = -log \frac{exp(sim(F_i, F_t)/\tau)}{\sum_{k=1}^{N} exp(sim(F_i, F_t)/\tau)} \quad (6)$$

Where $sim(\cdot)$ represents the cosine similarity function, τ is the temperature coefficient, and N is the number of samples in the batch. The final model optimization objective combines the spatiotemporal reconstruction loss and the contrast consistency constraint:

$$L_{total} = L_{recon} + \lambda L_{contrast} \quad (7)$$

Where L_{recon} represents the constraint term based on feature reconstruction, and λ is the weight balancing coefficient. Through this mechanism, the model achieves adaptive aggregation and discrimination of performance degradation patterns under unsupervised conditions, and can maintain stable representation consistency in complex distribution and dynamic dependency environments, thus providing a solid feature foundation for subsequent performance diagnosis and anomaly localization.

III. Experimental Analysis

A. Dataset

The dataset used in this study is the Alibaba Cloud AIOps Dataset, a real-world industrial dataset designed for research on microservice performance monitoring and anomaly analysis. It covers dozens of core microservice components from a large-scale online e-commerce platform and includes multidimensional performance metrics, trace logs, and system event records. The dataset spans a period of more than three months. Each microservice node provides high-frequency monitoring samples, including key operational indicators such as CPU usage, memory consumption, network latency, thread pool utilization, and request throughput. These detailed and multi-level features offer a rich foundation for modeling system performance degradation. The openness and high-dimensional structure of this dataset make it an essential experimental basis for self-supervised modeling and spatiotemporal feature extraction in microservice systems.

The structured design of the dataset captures the complexity of real-world microservice environments. Each sample record contains a timestamp, service instance identifier, metric vector, and associated topological dependency information. By constructing service dependency graphs, dynamic time-varying directed graphs can be formed to describe the evolution of invocation chains and resource contention among services. The dataset also provides multi-source feature dimensions, including application-level request logs, container-level performance metrics, and system-level resource states. This enables models to learn causal patterns of performance degradation across different granularities. In addition, the dataset defines performance states for different time windows, which facilitates the construction of sliding window sequences for self-supervised temporal modeling.

Based on this dataset, it is possible to evaluate the adaptive feature learning capability of models in complex topological environments. The performance degradation processes reflected in the dataset are often gradual rather than sudden, caused by combined factors such as network congestion, service dependency latency, and resource bottlenecks. Modeling and validation on this dataset can

 doi:10.20944/preprints202602.0480.v1

simulate how degradation propagates in real cloud-native microservice systems, providing a stable foundation for training self-supervised recognition models. The multidimensional, time-varying, and highly correlated characteristics of this dataset make it an important benchmark resource for research on performance degradation identification and spatiotemporal feature learning in large-scale microservice architectures.

B. Experimental Results

This paper first conducts a comparative experiment, and the experimental results are shown in Table 1.

Table 1. Comparative experimental results.

Method	Acc	Precision	Recall	F1-Score
XGBoost [8]	0.8712	0.8625	0.8481	0.8552
MLP [9]	0.8846	0.8763	0.8669	0.8715
1DCNN [10]	0.8928	0.8814	0.8742	0.8778
Transformer [11]	0.9043	0.8935	0.8891	0.8913
GNN [12]	0.9137	0.9062	0.8978	0.9019
SimCLR [13]	0.9211	0.9145	0.9107	0.9126
Ours	0.9468	0.9413	0.9386	0.9399

As shown in Table 1, the proposed self-supervised performance degradation identification model achieves the best results across all evaluation metrics in the comparative experiments. Traditional machine learning methods, such as XGBoost and MLP, show significantly lower accuracy, precision, and recall compared with deep learning-based models. This is because these traditional methods cannot effectively capture the dynamic dependencies in high-dimensional time series data of microservice systems. They usually rely on static feature inputs for decision-making and cannot adaptively learn the inter-service relationships and temporal evolution patterns. Therefore, their performance is limited when dealing with complex topological structures and non-stationary performance variations.

In contrast, deep neural network models such as 1DCNN, Transformer, and GNN improve the ability to capture degradation patterns by incorporating convolutional and graph-based modeling mechanisms. 1DCNN can recognize local feature fluctuations within short time windows, while Transformer and GNN further combine global attention with structural dependencies, demonstrating stronger spatiotemporal modeling capabilities. However, since these models still rely on supervised learning paradigms and lack semantic consistency constraints across time and services, their generalization and stability remain limited in dynamic microservice environments.

The performance improvement of self-supervised contrastive learning methods such as SimCLR indicates that unsupervised representation learning can effectively alleviate the limitations caused by label scarcity. By pulling similar samples closer and separating dissimilar ones in the feature space, the model can learn the underlying distribution of degradation patterns without explicit labels. This enhances the discriminability and robustness of learned features. These results confirm the applicability of self-supervised mechanisms in complex operations and maintenance scenarios, especially under high-dimensional and heterogeneous data conditions, where they outperform traditional supervised methods in feature learning.

Overall, the proposed model significantly outperforms all comparison methods in terms of accuracy, precision, recall, and F1-score. This demonstrates the effectiveness of the proposed spatiotemporal self-supervised framework for performance degradation identification. By integrating multi-source feature fusion, dynamic graph structure modeling, and self-supervised consistency constraints, the model captures the complex dependencies and temporal evolution patterns in microservice systems. The results show that the model not only achieves superior recognition accuracy but also exhibits strong generalization and stability, providing a reliable

 doi:10.20944/preprints202602.0480.v1

performance perception and early warning mechanism for intelligent operations in large-scale microservice environments.

The experimental results for different optimizers are also presented in Table 2.

Table 2. Experimental results of different optimizers.

Optimizer	Acc	Precision	Recall	F1-Score
AdaGrad	0.9176	0.9083	0.9014	0.9048
SGD	0.9262	0.9189	0.9127	0.9157
Adam	0.9375	0.9314	0.9273	0.9293
AdamW	0.9468	0.9413	0.9386	0.9399

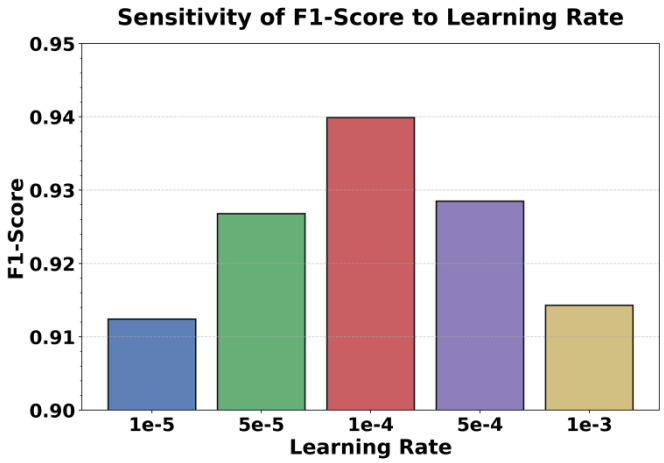
As shown in Table 2, different optimizers exhibit distinct differences in training stability and final model performance. The overall trend indicates that traditional gradient-based methods, such as AdaGrad and SGD, converge relatively quickly but are prone to getting stuck in local optima when dealing with complex spatiotemporal features and non-stationary distributions. This leads to noticeable fluctuations in precision and recall. AdaGrad suffers from rapid learning rate decay, which limits its ability to optimize fine-grained features in later stages. SGD shows more stable global convergence but lacks sensitivity in updating dynamic feature weights.

In contrast, Adam provides higher adaptability in parameter optimization through first and second moment correction mechanisms. It balances the gradient update speed across different feature dimensions, maintaining a stable convergence process in complex microservice datasets. In this study's self-supervised task, Adam performs better than traditional optimizers, especially in the F1-score, showing improved balance and indicating its superiority in handling data heterogeneity and temporal variation across multiple monitoring sources.

AdamW further introduces a weight decay mechanism, which helps prevent overfitting while maintaining convergence speed. This optimizer achieves the highest overall performance in accuracy, precision, and recall, demonstrating its ability to preserve feature distribution consistency and stability within the self-supervised learning framework. Its suppression of noisy gradients in high-dimensional feature spaces allows the model to capture latent temporal patterns of performance degradation more effectively, resulting in finer dynamic representations.

In summary, AdamW shows the best training stability and generalization capability in this task. This finding suggests that choosing optimizers with proper regularization can significantly enhance model robustness and spatiotemporal modeling effectiveness in large-scale microservice performance degradation identification. Its superior performance also provides a reliable optimization reference for deploying self-supervised identification models in complex industrial environments.

This paper also presents an experiment on the sensitivity of the learning rate to the F1-Score, and the experimental results are shown in Figure 2.



 doi:10.20944/preprints202602.0480.v1

Figure 2. Sensitivity experiment of learning rate to F1-Score.

As shown in Figure 2, different learning rates have a significant impact on the model's F1-score. This indicates that optimizer parameters play a crucial role in self-supervised performance degradation identification tasks. When the learning rate is too low (such as 1e-5), the model updates slowly and tends to fall into local optima. As a result, spatiotemporal features are not fully fitted, and the recognition performance is limited. When the learning rate increases moderately (up to 1e-4), the model can better capture the dynamic patterns of microservice performance degradation. It forms clearer decision boundaries in the feature space, and the F1-score reaches its highest value, suggesting that the spatiotemporal modeling and contrastive optimization processes are most stable.

When the learning rate continues to increase (such as 5e-4 or 1e-3), the model performance slightly declines. This is mainly due to unstable feature distributions caused by overly rapid gradient updates, which weaken the aggregation of degradation-related features. A higher learning rate causes oscillations during optimization, making it difficult for the model to maintain consistent representations in the self-supervised contrastive space. This reduces global feature consistency and local feature sensitivity. These results indicate that learning rate control is essential to ensure stable convergence when dealing with complex, multi-source monitoring data.

Overall, when the learning rate is around 1e-4, the model achieves the best balance between spatiotemporal representation and self-supervised optimization objectives. This reflects the adaptive advantage of self-supervised mechanisms when applied to multidimensional performance data. The results also verify that the model shows low sensitivity to hyperparameters in dynamic microservice scenarios. It can maintain stable performance degradation identification within an appropriate learning rate range, providing valuable guidance for parameter selection and automatic tuning in large-scale systems.

This paper also presents an experiment on the sensitivity of the residual coefficients to the F1-Score, and the experimental results are shown in Figure 3.

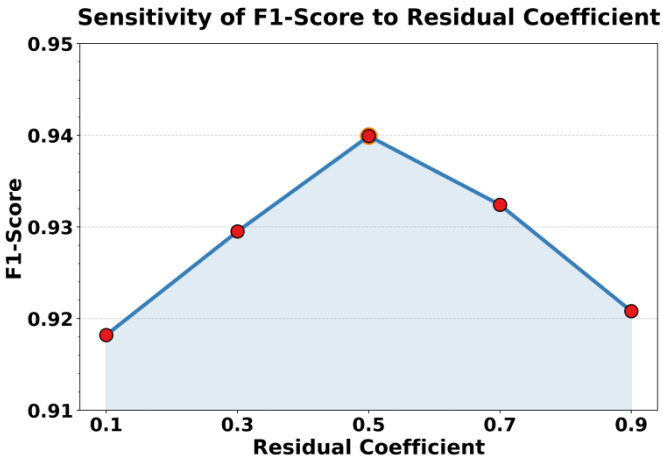


Figure 3. Sensitivity experiment of residual coefficients to F1-Score.

As shown in Figure 3, the residual coefficient has a significant impact on the model's F1-score, indicating that residual fusion plays a key regulatory role in self-supervised performance degradation identification tasks. When the residual coefficient is small (such as 0.1), the model mainly relies on representations from the current time step. This leads to insufficient information transfer and weak memory of historical dependencies, which limits the global aggregation of degradation features. As the residual coefficient increases, the model gradually enhances cross-time-step information fusion during feature updates. This improves the spatiotemporal consistency of the latent space, and the F1-score rises accordingly.

When the residual coefficient reaches a moderate level (such as 0.5), the model achieves peak performance. At this point, the residual branch and the main branch reach an optimal balance in

 doi:10.20944/preprints202602.0480.v1

information fusion. A moderate residual signal can preserve historical features while suppressing noise accumulation, improving the robustness and generalization of the model under complex dependency topologies. The model can more accurately capture latent performance relationships among microservices, making spatiotemporal feature representations more stable and supporting effective self-supervised aggregation and discrimination of degradation patterns.

When the residual coefficient continues to increase (such as 0.7 or higher), model performance begins to decline. This occurs because an overly strong residual signal weakens the dominance of the main feature update process, causing the model to rely too heavily on historical representations. As a result, representation redundancy and gradient imbalance appear under dynamic performance fluctuations. These findings suggest that in large-scale microservice environments, the design of residual mechanisms must maintain a dynamic balance between information transmission and feature innovation. Only by doing so can the self-supervised framework fully leverage its advantages in spatiotemporal feature modeling and achieve efficient and stable identification of performance degradation processes.

This paper also presents the effect of the temperature coefficient on the experimental results, and the experimental results are shown in Figure 4.

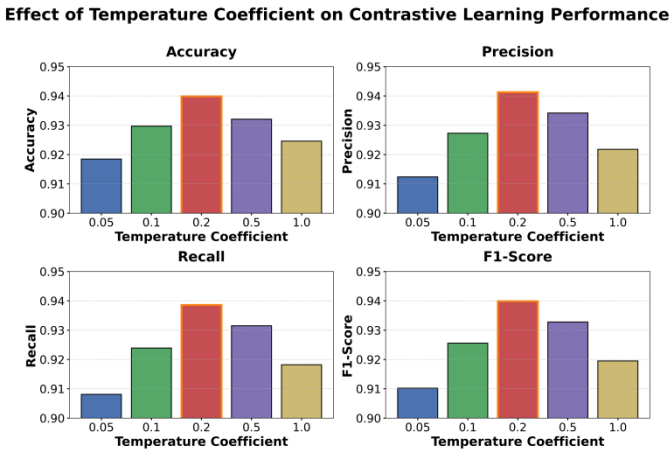


Figure 4. The effect of the temperature coefficient on experimental results.

As shown in Figure 4, the temperature coefficient in the contrastive learning framework has a significant impact on model performance. The temperature coefficient controls the separation scale between positive and negative samples, influencing how features are clustered and separated in the latent space. When the temperature coefficient is small (such as 0.05), the model pulls similar samples too close together, causing the feature distribution to become overly concentrated. This leads to unstable gradient updates and lower F1-score and other metrics. Although the model aggregates features quickly in this case, it struggles to distinguish between different degradation patterns, resulting in weak and less robust spatiotemporal feature boundaries.

When the temperature coefficient increases to a moderate range (such as 0.2), the model achieves peak performance across all metrics. At this level, the temperature balances the contrast strength between positive and negative samples, enabling the latent feature space to maintain intra-class consistency while preserving sufficient inter-class separability. This result shows that a moderate temperature setting can effectively enhance the discriminative capability of features in self-supervised contrastive learning, making the spatiotemporal representations clearer and more stable. This property is crucial for capturing fine-grained variations in microservice performance degradation and improves the model's sensitivity to degradation trends under complex dependency topologies.

When the temperature coefficient continues to increase (such as 0.5 or higher), model performance begins to decline slightly. A higher temperature weakens the distance constraints

 doi:10.20944/preprints202602.0480.v1

between positive and negative samples, causing features to become too dispersed in the latent space. This reduces the aggregation of degradation-related features. Under high temperatures, the model struggles to maintain stable alignment in feature contrast, leading to an imbalance between global feature consistency and local feature resolution, which results in decreased precision and recall.

Overall, the model performs best when the temperature coefficient is around 0.2. This indicates that in large-scale microservice performance degradation identification, the temperature setting in contrastive learning is a key factor affecting feature distribution stability and semantic aggregation capability. Proper temperature adjustment helps the model maintain information balance in high-dimensional feature spaces, allowing the self-supervised learning process to strengthen discriminative representations without excessive separation. This improves both the accuracy and generalization ability of performance degradation pattern recognition.

This paper also presents the impact of the monitoring sampling interval on the experimental results, and the experimental results are shown in Figure 5.

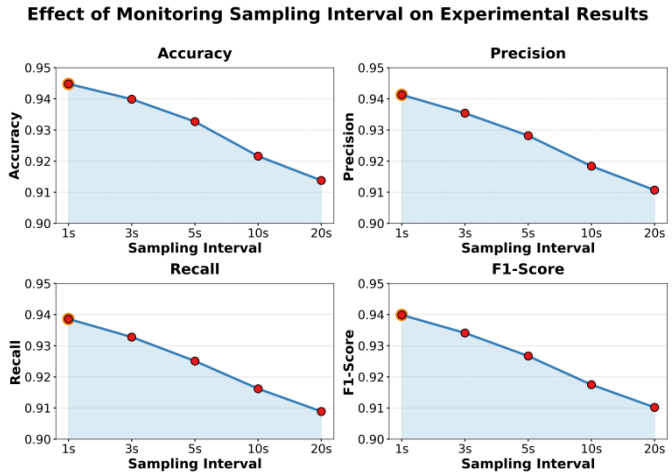


Figure 5. The impact of the monitoring sampling interval on experimental results.

As shown in Figure 5, the monitoring sampling interval has a clear impact on model performance. The four main metrics (Accuracy, Precision, Recall, and F1-score) all show a gradual decline as the sampling interval increases. When the sampling interval is short (such as 1s), the model can capture fine-grained system fluctuations and effectively learn short-term dependencies and sudden degradation patterns among microservices. At this stage, the model achieves the best overall accuracy and stability. The high temporal resolution of the data allows the model to better reconstruct the dynamic evolution of performance degradation, resulting in stronger spatiotemporal representation capability.

As the sampling interval increases (such as 3s or 5s), model performance gradually decreases. This suggests that longer sampling periods reduce the model's sensitivity to short-term anomalies and subtle fluctuations. Some transient degradation behaviors are smoothed out by the longer intervals, making it difficult for the model to capture fine-grained performance variations during training. As a result, feature representations in the latent space become more blurred, which lowers overall detection accuracy and recall. Although the model's training efficiency improves slightly in this setting, its ability to describe temporal degradation trends becomes clearly limited.

When the sampling interval is further extended to 10s or 20s, model performance declines significantly. Longer sampling intervals disrupt the temporal continuity of monitoring data, making it harder for the model to establish causal and sequential dependencies among services. Consequently, degradation patterns become diluted or obscured. In dynamic topologies and heterogeneous monitoring environments, long sampling intervals also amplify system noise and delay effects, reducing the model's responsiveness in highly dynamic workload scenarios.

 doi:10.20944/preprints202602.0480.v1

Overall, the experimental results show that the choice of sampling interval is crucial for performance degradation identification. Very short intervals provide richer information but increase monitoring costs and computational overhead, while overly long intervals significantly weaken spatiotemporal perception. The proposed model achieves its best performance at a 1s sampling interval, demonstrating its ability to fully capture the spatiotemporal dependencies of microservice performance degradation in high-frequency data environments. This provides valuable guidance for real-time performance analysis in intelligent operations and maintenance systems.

IV. Conclusion

This study proposes a self-supervised performance degradation identification model to address the complexity of identifying degradation in large-scale microservice architectures. The model integrates multi-source feature fusion, dynamic graph dependency modeling, and contrastive learning to achieve spatiotemporal consistency in system performance representation. Compared with traditional supervised detection methods, the proposed approach can effectively learn feature representations without labeled data. It fully explores hidden service dependencies and dynamic performance evolution patterns. Through the joint design of multidimensional feature representations and self-supervised optimization objectives, the model demonstrates strong adaptability and stability in both global structural modeling and local temporal perception, providing a new solution for identifying degradation under complex and non-stationary environments.

The experimental results show that the proposed model achieves significant improvements in accuracy, recall, and F1-score, confirming its superiority in capturing degradation trends in microservice systems. By introducing residual fusion and temperature regulation mechanisms, the model effectively balances structural consistency and feature discriminability, enabling efficient aggregation and separation of performance patterns in the latent space. Sensitivity analysis further reveals that learning rate, residual coefficient, temperature parameter, and sampling interval all have important impacts on model performance. These findings not only demonstrate the stability of the self-supervised mechanism in complex systems but also provide practical guidance for parameter optimization in different runtime environments.

This research has strong engineering significance. The proposed model can be directly applied to cloud platforms, container orchestration systems, and microservice monitoring frameworks to support early identification and trend prediction of performance anomalies. Its unsupervised nature allows continuous adaptive learning in multi-tenant and heterogeneous environments, reducing reliance on manual annotation and expert-defined rules. This contributes to building more resilient and intelligent distributed management systems. In addition, the model's scalable design provides a foundation for future self-diagnosis and automatic optimization, enabling resource scheduling, fault recovery, and performance tuning in complex operational scenarios.

Future work can further extend the applicability of the model to multimodal monitoring data by jointly modeling log texts, tracing data, and metric sequences to enhance the comprehensiveness of system state representation. It is also worth exploring the integration of self-supervised degradation identification with reinforcement learning strategies to achieve self-feedback optimization after anomaly detection, forming a closed-loop intelligent operations framework. Furthermore, as cloud-native and edge computing technologies continue to evolve, realizing distributed collaboration and knowledge transfer across hierarchical and cross-domain environments will be a key direction for advancing intelligent performance management toward autonomous computing. In summary, this research provides new theoretical insights and technical pathways for distributed system performance management and will have a profound impact on the development of future intelligent operation and maintenance systems.

References

 doi:10.20944/preprints202602.0480.v1

1. Ferreira R C, Trapmann R, van den Heuvel W J. MLOps with Microservices: A Case Study on the Maritime Domain[C]//Symposium and Summer School on Service-Oriented Computing. Cham: Springer Nature Switzerland, 2025: 3-15.
2. Santos W R M, Sampaio Jr A R, Rosa N S, et al. Microservices performance forecast using dynamic Multiple Predictor Systems[J]. Engineering Applications of Artificial Intelligence, 2024, 129: 107649.
3. 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.
4. Sun Y, Lin Z, Shi B, et al. Interpretable failure localization for microservice systems based on graph autoencoder[J]. ACM Transactions on Software Engineering and Methodology, 2025, 34(2): 1-28.
5. J. Jiang, C. Shao, C. Zhang, N. Lyu and Y. Ni, "Adaptive AI Spatiotemporal Modeling with Dependency Drift Awareness for Anomaly Detection in Large-Scale Clusters", 2025.
6. 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.
7. Khanahmadi M, Shameli-Sendi A, Jabbarifar M, et al. Detection of microservice-based software anomalies based on OpenTracing in cloud[J]. Software: Practice and Experience, 2023, 53(8): 1681-1699.
8. N. Lyu, Y. Wang, Z. Cheng, Q. Zhang and F. Chen, "Multi-Objective Adaptive Rate Limiting in Microservices Using Deep Reinforcement Learning", arXiv preprint arXiv:2511.03279, 2025.
9. Z. Zhang, Y. Xue, H. Zhu, S. Li, Z. Wang and Y. Xiao, "CondenseGraph: Communication-Efficient Distributed GNN Training via On-the-Fly Graph Condensation", arXiv preprint arXiv:2601.17774, 2026.
10. Iqbal A, Amin R. Time series forecasting and anomaly detection using deep learning[J]. Computers & Chemical Engineering, 2024, 182: 108560.
11. Sun Y, Guo Y, Liang M, et al. Multivariate time series anomaly detection based on pre-trained models with dual-attention mechanism[C]//2024 IEEE 35th International Symposium on Software Reliability Engineering Workshops (ISSREW). IEEE, 2024: 73-78.
12. Dutt A, Jang D, Nadkarni J, et al. GUIDE-GNN based Unified Incident Detection for Microservices Application Deployments[J].
13. Shang X, Zhang J, Jiang X, et al. Anomaly detection for multivariate time series based on contrastive learning and Autoformer[C]//2024 27th International Conference on Computer Supported Cooperative Work in Design (CSCWD). IEEE, 2024: 2614-2619.