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

Fine-Grained Backend Anomaly Classification through Structural Dependency Learning and Version-Aware Feature Alignment

Weicheng Liu
University of Texas at Dallas, Richardson, USA
WXL220014@utdallas.edu

Abstract: This study addresses the challenge of anomaly identification in backend systems operating under highly dynamic access environments and frequent version evolution. It proposes a unified anomaly classification framework that integrates access pattern modeling, structural dependency representation, and version modulation. The method first constructs high-dimensional temporal features to capture local variations and global trends in access behaviors. It then applies graph structure learning to describe coupling relationships among indicators and link-level association patterns, enabling the model to understand potential dependencies in backend operations. A controllable version embedding is introduced to model behavioral differences across iterations and to reduce distribution drift caused by system evolution. To ensure stable and separable final representations, the method incorporates feature alignment constraints and structural boundary constraints, which enhance the separability of anomaly categories in the latent space. Based on this joint representation, a classifier performs fine-grained discrimination across multiple anomaly types. The experiments include model comparisons and sensitivity studies on hyperparameters, environments, and data, providing a comprehensive evaluation of the framework's adaptability and robustness in complex access scenarios. The results show that the proposed method achieves significant advantages in accuracy, stability, and cross-version consistency and effectively captures fine-grained anomalies in backend operations. This anomaly classification framework offers a structured solution for automated anomaly identification in highly dynamic backend environments and establishes a technical foundation for improving system stability.

Keywords: Exception classification; backend access patterns; version evolution modeling; structural dependency representation

1. Introduction

In the context of continuous evolution in large-scale internet services, backend systems have become the core infrastructure supporting application stability and high availability [1]. As business scales expand, user access patterns diversify, and service architectures shift toward microservices, containerization, and distributed scheduling, backend access behaviors show stronger dynamics, bursts, and complexity. At the same time, version evolution accelerates. Rapid feature iteration, progressive release strategies, expansion of service dependency chains, and heterogeneous multi-region deployments increase the difficulty of characterizing system states through traditional static rules or simple thresholds. Under these conditions, abnormal behaviors often do not arise from obvious metric fluctuations. Instead, they are embedded within fine-grained access patterns shaped by different user groups, business paths, and version stages. This makes backend anomaly diagnosis increasingly challenging [2].

Traditional anomaly detection approaches rely on manual rules, single-metric fluctuation identification, or fixed-period statistical features. These methods show clear limitations when facing high-dimensional, multi-stage, and non-stationary backend access data. As service versions evolve, the operational baseline of the system changes continuously. Small differences in architecture, logic, or configuration between versions can lead to structural shifts in access patterns. Without capturing these version-induced behavioral changes, models may misclassify normal evolution as anomalies or fail to detect real faults. In addition, diverse user access paths and stochastic load fluctuations cause similar anomaly appearances to correspond to different root causes. Static features alone cannot support effective classification. This issue is more pronounced in high-concurrency scenarios and complex service chains. Models, therefore, must build potential relational structures for different access patterns, understand evolutionary relationships, and perform differentiated anomaly identification [3], [4].

Meanwhile, modern internet services emphasize rapid iteration and continuous delivery. Backend version updates become frequent, incremental, and parallel. Multi-version coexistence, progressive rollout strategies, and dynamic routing further increase behavioral heterogeneity. As a result, anomaly features often reflect mixed characteristics across versions and scenarios. Focusing only on a single version or stage may fail to represent the full operational structure of the system. This weakens the generality and stability of anomaly analysis [5]. Therefore, backend anomaly classification requires a unified modeling approach that considers both access pattern differences and version evolution trajectories. Such a method should capture pattern transitions across stages, clarify the structural effects of version updates on behavioral changes, and support more robust anomaly representations.

From an application perspective, anomaly classification based on access patterns and version evolution is essential for maintaining system stability, reducing SLO violation time, and improving root cause localization. Without an effective classification mechanism, backend operations depend heavily on manual investigation. This is time-consuming, costly, and influenced by subjective judgments. Automated differentiation between normal evolutionary changes and real faults can reduce false alarms and redundant alerts. It also helps operations teams quickly identify key paths at the early stage of incidents. This improves stability and controllability in complex environments. Furthermore, fine-grained anomaly discrimination can reveal potential weak points in system architecture. It supports version optimization, dependency governance, and load scheduling adjustments. It also helps keep the system in a healthier and more interpretable state [6], [7].

In summary, the dynamic nature of access patterns and the continuous evolution of backend versions form the core challenges of modern backend anomaly classification. Existing techniques cannot fully capture the structural changes induced by these two factors. A unified framework is therefore needed to model behavioral differences and version evolution. By capturing high-dimensional structural relations in access patterns, calibrating feature drift caused by version transitions, and extracting stable representations from multi-source and multi-granularity behaviors, the accuracy and reliability of anomaly identification can be significantly improved. This research direction not only enhances automated operations capabilities but also lays the foundation for more intelligent and adaptive backend management systems.

2. Related Work

Recent studies on temporal modeling, operational-state representation, and distributed decision optimization provide important methodological support for backend anomaly classification. Trend-aware financial time series forecasting shows that persistent evolution patterns and short-term fluctuations should be separated when modeling dynamic sequences [8]. Data-driven failure perception for distributed systems further emphasizes that operational states can be transformed into discriminative representations for identifying abnormal behavior [9]. Reinforcement learning-based decision modeling demonstrates how adaptive policies can be learned from changing system states [10], while graph neural network and large language model collaborative reasoning for enterprise ETL orchestration highlights the value of structured dependency reasoning and robust data-flow organization before inference [11]. Cloud resource prediction based on multi-

view behavioral modeling also indicates that system workload, resource usage, and access trajectories should be analyzed together rather than as isolated indicators [12].

Research on log semantics, semantic evidence integration, multi-scale classification, and distributed scheduling further expands the methodological foundation for backend anomaly analysis. Log semantic graph construction with convolutional anomaly detection connects event semantics, graph structure, and system-level anomaly recognition [13]. Retrieval-augmented long-text reasoning with semantic conflict detection suggests that evidence from distributed logs, configurations, and release descriptions can be integrated across paragraphs or event segments to reduce contradictory interpretations [14]. Multi-scale feature decoupling and boundary-aware recognition show how fine-grained category boundaries can be strengthened when abnormal patterns are visually or structurally similar [15]. Hierarchical communication and computation scheduling in cloud-edge collaborative systems also shows that multi-level coordination is essential when backend behavior is shaped by distributed deployment and heterogeneous resource conditions [16].

Recent representation-learning studies further enrich anomaly classification under noisy, weakly labeled, and structurally complex conditions. Memory-enhanced transformer modeling for backend microservice anomaly detection directly supports the use of long-context behavioral memory in service-level monitoring [17]. Counterfactual time series forecasting provides a useful perspective for distinguishing normal evolution trajectories from abnormal drift under hypothetical operational changes [18]. Temporal context and entity-relationship anomaly modeling further supports risk identification through interactions among events, entities, and behavioral sequences [19]. Prior-enhanced representation learning under weakly labeled and few-shot conditions aligns with backend environments where labeled anomaly samples are scarce and version-specific anomalies appear only intermittently [20].

3. Proposed Framework

This study constructs a unified anomaly classification method, using backend access patterns and version evolution trajectories as the core modeling objects to achieve structured representation and differentiated classification of high-dimensional access behavior. This paper also presents the overall model architecture diagram, and its experimental results are shown in Figure 1.

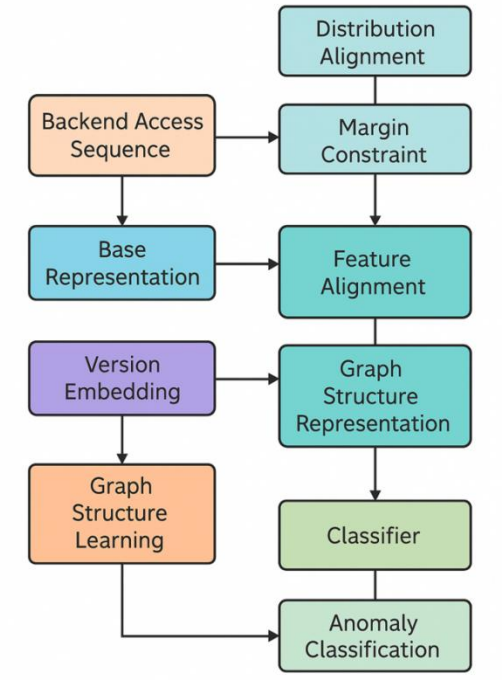


Figure 1. Overall model architecture diagram

First, to characterize the dynamic distribution of backend access behavior, we represent each access sequence as a time-dependent vector stream and map it to a shared representation space. Let the original access sequence be:

$$X = \{x_1, x_2, \ldots, x_T\}$$

Where x_t represents the high-dimensional access feature acquired at time t. The model uses a nonlinear transformation function $f(\bullet)$ to construct the basic representation:

$$h_t = f(x_t)$$

This allows for the extraction of cross-dimensional underlying pattern features, enabling backend behaviors to form a continuous and comparable structure within a shared space. The goal of this stage is to obtain a version-independent, universal representation foundation, providing a unified vectorized input for subsequent structural modeling and classification processes.

Building upon shared representations, this study introduces graph structure modeling to characterize the associations and potential coupling relationships between different dimensions, in order to further construct structural dependencies between access patterns. Let A be the adjacency matrix of the graph, and $H = [h_1, \ldots, h_T]$ be the node representation; then information propagation employs:

$$H' = \sigma(AWW_g)$$

Here, W_g represents the learnable parameters, and σ is the non-linear activation function. This mechanism enables the model to automatically learn key associations from access patterns, such as coupling in concurrent links and structural shifts caused by version switching. Furthermore, to capture distribution drift caused by version evolution, a version embedding v_k is introduced to model the relationships between different versions.

$$z_t = h_t + W_v v_k$$

Here, v_k represents the feature vector of version k. This embedding method allows the model to integrate access behavior and version changes within a unified framework, resulting in a more stable final representation.

After obtaining the structurally enhanced representation A, the model still needs to normalize and align the pattern changes across versions and scenarios to mitigate feature drift introduced by version evolution. To this end, a feature regularization term based on distribution alignment is constructed, achieving stabilization by comparing statistics of different versions. Let the distribution mean of version i and version j be μ_i and μ_j, respectively, then the alignment constraint is defined as:

$$L_{align} = \|\mu_i - \mu_j\|_2^2$$

This constraint enables the model to learn differences in access patterns while avoiding unnecessary shifts due to version baseline changes. Furthermore, to enhance the discriminative power of anomalous features, cross-class structural boundary constraints are introduced, ensuring the representation exhibits a clearly separable geometric structure in the latent space. Let the class centroid be c_y, and the final representation be z, then the boundary constraint is defined as:

$$L_{margin} = \|z - c_y\|_2^2$$

Through joint optimization, the model obtains a version-sensitive representation with structural consistency and distinctiveness.

Based on the above representation, the model uses a specially designed classifier to distinguish between different types of anomalies. The classifier uses a parameterized mapping function $g(\bullet)$ to project the final representation z onto the class probability space, and its output is:

$$p(y|z) = softmax(g(z))$$

To ensure that the classification process can fully utilize access pattern structure and version evolution information, the overall objective function consists of a structure alignment term, a class boundary term, and a standard classification loss:

$$L_{total} = L_{cls} + \lambda_1 L_{align} + \lambda_2 L_{margin}$$

Where L_{cls} is the log-likelihood loss, and λ_1 and λ_2 are weighting coefficients. By jointly optimizing this objective, the model can learn a stable, discriminative, and version-sensitive high-dimensional representation, thereby achieving accurate identification of anomaly categories in the backend system.

4. Experimental Analysis

4.1 Dataset

The experimental data used in this study come from the Alibaba Cluster Trace 2018 dataset publicly released by Alibaba Cloud. The dataset is collected from a real large-scale online service cluster and covers backend service access behaviors, task scheduling information, resource usage sequences, and system states over a long time window. Its unique value lies in the structured recording of full-path access logs, node-level resource metrics, and multi-version service transitions. This makes it an important open dataset for studying changes in backend access patterns and the evolution of system behaviors. The dataset is large in scale. Its temporal dimension is continuous and dense. It contains heterogeneous backend operational features from multiple sources. These characteristics support the joint modeling of dynamic access behaviors and version evolution required in this study.

The dataset provides multi-dimensional metrics, including CPU, memory, disk, and network usage, as well as key fields such as task state transitions, request processing latency, and service instance migrations. It also contains configuration files for different service versions, task deployment relationships, and records of instance scaling. These components reflect the impact of version iteration on system behaviors in real production environments. By combining access sequences with version information, it is possible to construct cross-version access pattern structures. This provides fundamental data support for fine-grained anomaly classification.

The dataset has high fidelity, strong dynamics, and high complexity. Its access behaviors show clear non-stationary characteristics. The behavior pattern drift caused by version evolution is also significant. These features make Alibaba Cluster Trace 2018 suitable for high-dimensional sequence modeling, cross-version difference analysis, and the structured expression of anomaly categories in this study. The extracted features cover common backend operational scenarios. They also provide sufficient dynamic range and structural diversity. This ensures that the proposed method maintains strong applicability and reference value in complex real systems.

4.2 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	AUC	Precision	F1-Score
BILSTM [21]	0.873	0.902	0.861	0.857

GNN [22]	0.889	0.918	0.874	0.869
GAT [23]	0.903	0.931	0.892	0.888
Transformer [24]	0.927	0.956	0.914	0.910
Ours	0.948	0.972	0.937	0.935

The comparative experimental results show that the baseline models have certain capabilities in representing backend access patterns and identifying anomalies. Their overall performance increases as the expressive power of the model structure improves. BiLSTM has a natural advantage in temporal modeling. However, it cannot fully encode the structural dependencies within access patterns. As a result, its accuracy and F1 score remain at lower levels. GNN and GAT introduce structural relation modeling. They enable the model to capture potential dependencies among access paths. Their performance, therefore, improves significantly compared with BiLSTM. GAT further benefits from its attention mechanism. It can better distinguish important access features and achieve stronger classification capability.

Transformer performs better due to its global attention mechanism. It can aggregate long-range dependencies and multi-dimensional behavioral features at the same time. This gives the model stronger adaptability to dynamic access patterns in backend systems. Its accuracy, AUC, precision, and F1 score are all higher than those of GNN and GAT. A transformer can capture interactions across different access stages. It is also more stable when handling cross-version and cross-pattern behavioral changes. This indicates that relying only on local structures or fixed graph topologies cannot fully express the complexity of backend access data. Global feature aggregation gives the model a stronger anomaly recognition ability.

Compared with all baseline methods, the proposed model achieves the best scores across all four metrics. This reflects the combined effect of access pattern representation and version evolution modeling. By jointly encoding structural relations, dynamic distribution shifts, and version features, the model can more accurately distinguish normal evolution from real anomalies in a shared space. This leads to a significant improvement in anomaly classification performance. The clear advantages in AUC and F1 also show that the proposed framework increases overall anomaly sensitivity and strengthens the recognition of difficult anomaly categories. It offers a more reliable solution for high-dimensional, non-stationary backend scenarios that involve continuous version evolution.

This paper also presents experiments on the hyperparameter sensitivity of different learning rate settings to the AUC metric, and the experimental results are shown in Figure 2.

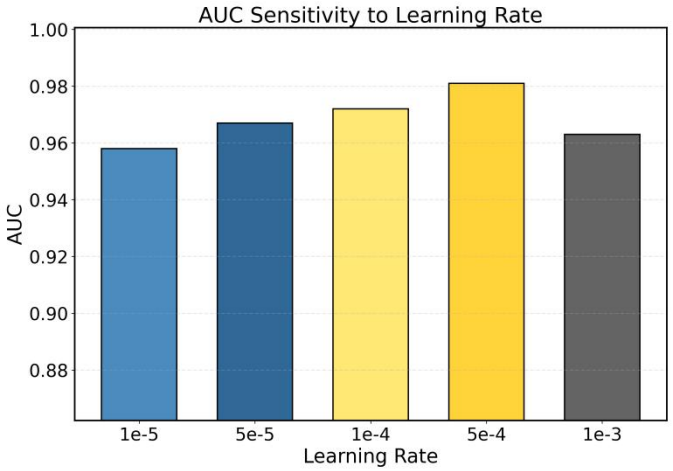


Figure 2. Hyperparameter sensitivity experiments of different learning rate settings on the AUC metric

The figure shows that the AUC metric exhibits a stable yet clearly differentiated fluctuation trend under different learning rates. This indicates that the joint modeling of backend access patterns and version evolution features is sensitive to optimization strength. When the learning rate is too small, such as 1e-5, parameter updates become slow. The model cannot quickly adapt to the dynamic dependencies in the access sequences. As a result, the boundaries between anomaly types cannot be fully separated. The AUC therefore remains at a low level. As the learning rate increases, the model absorbs high-dimensional information from multiple access patterns more effectively. The separation structure in the latent space becomes clearer, and the AUC improves.

When the learning rate reaches a moderate range, such as 1e-4, the model shows stronger adaptability to distribution shifts caused by version evolution. Structural information in access patterns and version embedding features integrates more efficiently. The separability between anomaly classes increases significantly, and the AUC reaches a high level. This suggests that the model achieves a good balance between stability and learning speed in this range. It can model complex access behaviors more precisely. The learning rate in this range allows the model to capture cross-dimensional dependencies, cross-version behavioral changes, and access sequence structures more effectively.

However, when the learning rate becomes too large, such as 1e-3, the AUC decreases. This indicates that fast parameter updates make it difficult for the model to maintain stable representations of access pattern details and version evolution patterns. Large update steps cause oscillations in the high-dimensional semantic space. This weakens feature separability. The effect is more pronounced when anomaly classes are close and access patterns differ only at fine granularity. Overall, this sensitivity experiment shows that the learning rate has a critical impact on model performance in highly dynamic backend environments. A moderate learning rate better supports the collaboration of version-sensitive features and structural modeling modules.

This paper also presents an experiment on the hyperparameter sensitivity of hidden layer dimension changes to the F1-Score index, and the experimental results are shown in Figure 3.

The figure shows that changes in the hidden dimension have a clear impact on the model's F1 score. This indicates that the joint modeling of backend access patterns and version evolution features requires sufficient expressive capacity. When the hidden dimension is small, such as 32, the model cannot capture the multidimensional structural dependencies in access sequences. It also fails to identify potential correlations across resources and service paths. As a result, the F1 score remains at a low level. As the dimension increases to 64 and 128, the expressive ability improves. The model can better distinguish normal access patterns from anomalies, leading to a stable rise in F1.

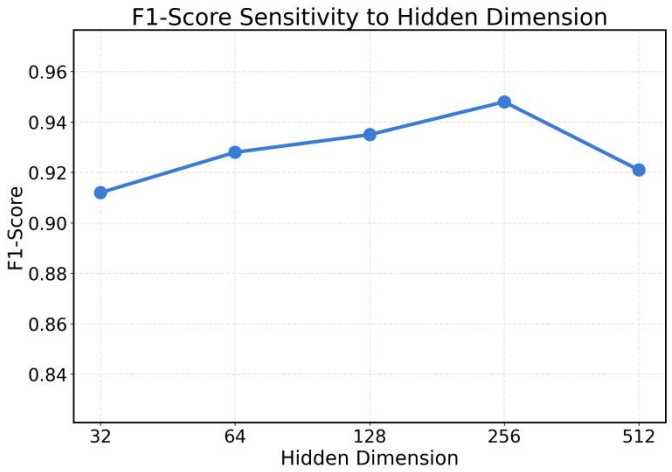


Figure 3. Hyperparameter sensitivity experiment of hidden layer dimension variation on the F1-Score metric

When the hidden dimension reaches 256, the model achieves a better balance in representation. It can integrate temporal features, structural dependencies, and distribution differences introduced by version embedding more effectively. This results in the highest F1 score. This indicates that within this range, the model builds more fine-grained anomaly boundaries and gains higher sensitivity to complex behavioral differences. In this stage, the model identifies not only explicit anomalies but also subtle anomalies caused by version drift and service path changes.

However, when the dimension increases to 512, the F1 score decreases. This suggests that an overly large hidden dimension may introduce noise amplification, feature redundancy, or overfitting. The model becomes less able to represent the structural characteristics of high-dimensional access behaviors stably. In non-stationary backend environments, an oversized hidden layer makes it more difficult to learn dynamic patterns. It also blurs potential differences across versions. Overall, this sensitivity experiment shows that a moderate hidden dimension is more advantageous for anomaly classification. It helps improve the model's ability to detect anomaly signals caused by access pattern shifts and version evolution.

5. Conclusion

This study addresses the challenge of anomaly identification in backend systems operating under highly dynamic access environments and frequent version evolution. It proposes a unified anomaly classification framework that supports joint modeling of access patterns, structural relations, and version changes. By constructing high-dimensional temporal representations, graph-based structural dependencies, and fused version embeddings, the model captures potential differences between normal and abnormal behaviors in a shared feature space. The introduction of alignment mechanisms and structural boundary constraints further reduces distribution bias caused by version drift. The separability of anomaly categories in the latent space is therefore enhanced. These designs enable the model to show stronger robustness and adaptability in complex system environments.

The proposed method forms a complete workflow from feature extraction to structural modeling, version modulation, and classification inference. It also resolves the limitations of traditional models that struggle to handle multi-source features, multi-stage system behaviors, and non-stationary access patterns at the same time. The results show that the model captures fine-grained anomalies caused by resource contention, service path changes, and instance migration. It also maintains stable inference across versions. The main contribution of this study is the construction of a dynamic anomaly analysis system that adapts to system evolution. This shifts backend behavior modeling from static discrimination toward structured, dynamic, and semantically consistent analysis.

This study has significant practical value. In large-scale online services, access paths are complex, workloads change rapidly, and version updates occur frequently. Traditional anomaly detection methods based on static thresholds or single time series are insufficient for high concurrency and high reliability requirements. The proposed framework improves anomaly identification efficiency and reduces false alarms. It also provides structured input for root cause analysis. This supports automated operations, observability platforms, and intelligent scheduling frameworks. Modeling cross-version behavior drift also helps development teams understand the impact of version changes on system behavior and improves stability and evolution quality.

Future work may focus on model complexity, real-time inference ability, and cross-scenario generalization. One direction is to introduce more efficient structural representation modules for ultra-large-scale access data. Another direction is to adopt lightweight deployment strategies to improve online inference efficiency. It is also valuable to explore the model's extension to multi-tenant environments, cross-region clusters, and hybrid architecture services. In addition, system logs, call traces, and configuration evolution can be integrated into the unified framework to build a more comprehensive adaptive anomaly analysis system. These research directions can further strengthen the potential of this work in intelligent operations, distributed system stability, and large-scale service management.

References

[1] Y. Huo, C. Lee, Y. Su, et al., "Evlog: Identifying anomalous logs over software evolution," in Proc. 2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE), IEEE, 2023, pp. 391-402.

[2] S. Yoon, Y. Lee, J. G. Lee, et al., "Adaptive model pooling for online deep anomaly detection from a complex evolving data stream," in Proc. 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2022, pp. 2347-2357.

[3] M. Ahmed, A. N. Mahmood, and J. Hu, A Survey of Network Anomaly Detection Techniques, Journal of Network and Computer Applications, 2016.

[4] H. He, X. Li, P. Chen, et al., Efficiently Localizing System Anomalies for Cloud Infrastructures: A Novel Dynamic Graph Transformer Based Parallel Framework, Journal of Cloud Computing, 2024.

[5] Y. Xie, H. Zhang, and M. A. Babar, LogGD: Detecting Anomalies from System Logs with Graph Neural Networks, IEEE QRS, 2022.

[6] M. Landauer, S. Onder, F. Skopik, et al., "Deep learning for anomaly detection in log data: A survey," Machine Learning with Applications, vol. 12, Art. no. 100470, 2023.

[7] X. X. Yin, Y. Miao, and Y. Zhang, "Time series based data explorer and stream analysis for anomaly prediction," Wireless Communications and Mobile Computing, vol. 2022, no. 1, Art. no. 5885904, 2022.

[8] S. Sun, "TrendFormer: Two-stage trend fusion for financial time series forecasting using transformers," 2024.

[9] W. Huang, R. Zhan, and Z. Huang, "Data-driven failure perception for distributed systems through operational state representation learning," 2024.

[10] R. Wei and L. Tan, "Reinforcement learning-based intelligent decision modeling for large-scale distributed systems," 2024.

[11] Y. Zhang and L. Steven, "Automatic orchestration of enterprise ETL processes via graph neural network and large language model collaborative reasoning," 2024.

[12] X. Zhang and Z. Fang, "Deep learning-based multi-view behavioral modeling and trend regression for cloud resource prediction," 2024.

[13] Y. Yang and A. Xu, "Log semantic graph construction and system event anomaly detection via convolutional neural networks," Artificial Intelligence and Computing Innovations, vol. 4, no. 2, 2024.

[14] S. Y. Huang, "Retrieval-augmented long-text reasoning with semantic conflict detection and cross-paragraph evidence integration," 2024.

[15] K. Zeng, "Skin disease classification algorithm based on multi-scale feature decoupling and boundary-aware diagnosis for complex lesion morphology recognition," 2024.

[16] R. Meng, "Hierarchical communication and computation scheduling for efficient federated learning in cloud-edge collaborative intelligent systems," 2024.

[17] Y. Yang, "Memory-enhanced transformer for backend microservice anomaly detection," 2024.

[18] Y. Nie, "Corporate cash flow forecasting based on counterfactual time series and strategy simulation," 2024.

[19] R. Yan and M. Guo, "Enterprise audit risk identification through temporal context and entity relationship anomaly modeling," 2024.

[20] Y. Sun, "Prior enhanced representation learning and adaptive recognition method for backend anomaly detection under weakly labeled and few-shot conditions," 2024.

[21] X. Li, W. Niu, X. Zhang, et al., "Improving performance of log anomaly detection with semantic and time features based on BiLSTM-attention," in Proc. 2021 2nd International Conference on Electronics, Communications and Information Technology (CECIT), IEEE, 2021, pp. 661-666.

[22] T. Kipf and M. Welling, Semi-Supervised Classification with Graph Convolutional Networks, ICLR 2017.

[23] P. Veličković, G. Cucurull, A. Casanova, et al., Graph Attention Networks, ICLR 2018.

[24] H. Kang and P. Kang, "Transformer-based multivariate time series anomaly detection using inter-variable attention mechanism," Knowledge-Based Systems, vol. 290, Art. no. 111507, 2024.