Journal of Computer Technology and Software

ISSN:2998-2383

Vol. 3, No. 9, 2024

Intelligent Defect Detection and Risk Assessment for Cloud Platforms Using Counterfactual System Modeling

Chengda Xu
University of Washington, Seattle, USA
williamxu026@gmail.com

Abstract: This study addresses the challenges of hidden defect behavior, complex causes, and strong sensitivity to operating condition changes in cloud computing environments. It proposes a defect detection method oriented toward understanding system operation mechanisms. The method is based on multi-source monitoring data and models system operation as continuously evolving state sequences. System behavior deviation is characterized through state prediction and consistency analysis. Conditional intervention modeling is introduced within a unified framework to construct alternative operational scenarios and evaluate the impact of key condition changes on system behavior. By jointly modeling observed states and state responses under condition changes, the method highlights core factors that are highly relevant to defect discrimination in complex dynamic environments. It reduces misjudgment risk caused by noise interference and incidental fluctuations. Under a unified data setting, comparative analysis with several representative methods shows that the proposed approach achieves more consistent performance in overall discrimination stability and risk differentiation capability. This work provides a mechanism-aware modeling perspective for defect identification and risk analysis in complex cloud platforms. It contributes to a deeper understanding and more reliable analysis of system anomalies in intelligent operations scenarios.

Keywords: Causal-driven modeling; operational status characterization; risk quantification and assessment; intelligent operation and maintenance analysis

1. Introduction

With the rapid evolution of cloud computing infrastructure, computing resources are expanding from centralized data centers toward large-scale, cross-regional, virtualized, and service-oriented environments. These infrastructures support critical application domains such as finance, government services, industrial internet, and artificial intelligence. Through mechanisms including resource pooling, elastic scheduling, and multi-tenant sharing, cloud systems achieve high utilization and scalability. At the same time, they introduce highly complex system structures and operating conditions. Hardware heterogeneity, layered virtualization, decoupled service components, and dynamic workloads lead to system states with strong nonlinearity and tight coupling. Under such conditions, defects in cloud systems rarely appear as isolated and easily localized failures. They instead emerge gradually as performance fluctuations, service degradation, abnormal call chains, or implicit logical errors, posing latent risks to service reliability and business continuity[1].

From a mechanistic perspective, defects in cloud computing systems exhibit clear characteristics of multi-source origin and temporal evolution. On the one hand, logical-level issues such as resource contention, scheduling conflicts, and configuration drift may accumulate over long periods before triggering abnormal behavior. On the other hand, environmental disturbances such as network instability, node failures, or sudden workload changes can propagate and amplify through service dependencies. Because service instances in cloud environments are highly dynamic, system states continuously evolve [2]. Defects are therefore difficult to characterize using static rules or single indicators. This high degree of dynamics and uncertainty limits the effectiveness of traditional detection approaches based on fixed thresholds, empirical rules, or post-event analysis, especially in complex cloud scenarios[3].

In recent years, intelligent methods have shown strong potential in system operation and defect detection by modeling multi-source data such as logs, metrics, and call traces. These approaches can capture system behavior patterns at the data level. However, most existing methods focus on correlation modeling and pattern matching. They primarily address whether an anomaly occurs. Their ability to explain why it occurs or whether it would still occur under different conditions remains limited. In cloud systems, numerous causal relationships exist among components. Methods based solely on correlation often fail to distinguish true defect drivers from accompanying phenomena[4]. This limitation can lead to misjudgment or weak interpretability under complex interference. The lack of causal awareness reduces the practical value of detection results for system optimization and risk prevention.

Counterfactual reasoning offers a new perspective for addressing these challenges. By constructing hypothetical scenarios that ask how system states would evolve if certain

conditions were changed, counterfactual modeling explicitly captures system sensitivity to key factors. In cloud computing systems, the introduction of counterfactual architectures helps overcome the limitations of passive judgment based only on observed data[5]. Defect detection can thus move beyond outcome identification toward mechanism understanding. By comparing actual execution trajectories with system responses under counterfactual conditions, it becomes possible to identify critical conditions that lead to defects. This process also helps distinguish incidental disturbances from structural risks, providing stronger evidence for defect analysis in complex systems.

For these reasons, research on cloud computing system defect detection that integrates counterfactual architectures has important theoretical and practical significance. At the theoretical level, it promotes a shift from correlation-driven detection to causality-aware analysis and deepens understanding of complex system behavior[6]. At the methodological level, counterfactual modeling provides a unified framework for describing potential system evolution under different assumptions, which improves robustness and interpretability. At the application level, defect detection mechanisms with counterfactual analysis capability can better support operational decision making and risk governance in cloud platforms. This capability strengthens the long-term reliability and security of cloud systems in high-assurance application scenarios.

2. Background

Modern cloud computing systems have evolved into highly dynamic and heterogeneous infrastructures that integrate distributed scheduling, multi-tenant resource sharing, and service-oriented architectures. As system scale continues to expand, efficient resource management and adaptive scheduling have become increasingly critical challenges. Early studies explored optimization through reinforcement learning-based policies, where dynamic cache replacement and scheduling decisions are learned from continuous interaction with system environments [7]. Meanwhile, causal inference theory provides a principled framework for understanding system behavior beyond simple correlations, enabling reasoning about intervention effects and structural dependencies [8]. These perspectives have driven recent efforts to combine learning-based scheduling with causality-aware modeling in distributed systems.

Recent advances in intelligent scheduling frameworks further emphasize semantic representation and policy generation. Semantic-driven scheduling methods leverage unified representations to bridge system states and decision policies, thereby improving adaptability in complex distributed environments [9]. In parallel, counterfactual reasoning has been extensively studied in large-scale decision systems such as computational advertising, where learning under hypothetical interventions enhances robustness and decision quality [10]. Building upon these ideas, multi-agent reinforcement learning integrated with graph-based modeling has been proposed to capture inter-component dependencies and enable cooperative resource scheduling across distributed systems [11]. Additionally, hierarchical agent architectures incorporating large language model reasoning have demonstrated strong capabilities in complex task planning and dynamic decision-making [12].

From a system-level perspective, real-world cloud environments exhibit highly dynamic workload patterns and complex resource interactions. Large-scale datasets, such as Google cluster traces, provide empirical evidence of workload heterogeneity, temporal variability, and scheduling complexity [13]. These characteristics introduce significant challenges for reliability and anomaly detection, particularly when system behavior is influenced by multiple interacting factors. To improve decision transparency and controllability, recent work on algorithmic recourse emphasizes actionable interventions derived from counterfactual reasoning, enabling systems not only to detect anomalies but also to suggest corrective actions [14].

In intelligent system monitoring and anomaly detection, large language models and generative AI frameworks have recently been introduced to enhance semantic understanding and risk reasoning. For instance, LLM-based frameworks have been applied to financial anomaly detection by integrating multi-document reasoning and semantic mapping [15]. In cloud systems, generative modeling approaches combined with uncertainty quantification improve anomaly detection robustness under noisy and non-stationary conditions [16]. At the system level, resource scheduling and anomaly detection are inherently coupled problems. Classical scheduling approaches, such as multi-resource packing, focus on utilization efficiency but lack adaptability to dynamic system states [17]. Meanwhile, explainable AI methods have been explored in domains such as healthcare to improve interpretability and trustworthiness of decision-making systems [18].

To better model complex system environments, recent studies incorporate semantic priors and structured representations into AI frameworks, enabling more robust decision-making under uncertainty [19]. Distributed anomaly detection architectures for microservice systems further highlight the importance of capturing cross-service dependencies and propagation patterns [20]. In addition, data distribution shifts and class imbalance remain key challenges in real-world systems. Drift-aware adaptive classification methods dynamically adjust model behavior under evolving environments [21]. Representation learning techniques, particularly those based on multi-task self-supervision, have also demonstrated strong capability in modeling spatiotemporal dependencies in complex time-series data [22].

Log-based and metric-based anomaly detection remain fundamental components of cloud system monitoring. Early work on system log analysis demonstrated the feasibility of extracting anomaly patterns from large-scale logs [23], while recent surveys summarize the evolution of root cause analysis techniques in microservice architectures [24]. Modern approaches further integrate statistical learning and deep learning to improve detection accuracy in distributed systems [25], including log-based anomaly detection without explicit parsing [26]. In parallel, multimodal and attention-based

reasoning methods have been explored to enhance decision consistency in complex domains such as financial reasoning [27] and multimodal perception tasks [28].

Causal modeling has also been widely applied in recommendation and decision systems, where exposure bias and structural dependencies significantly affect model performance [29]. Privacy-preserving and federated learning approaches extend these ideas by enabling collaborative optimization under data constraints [30]. In time-series modeling, graph neural networks and transformer-based architectures have been used to capture complex temporal and structural dependencies in large-scale systems [31], which are particularly relevant for anomaly detection in dynamic cloud environments.

Deep learning-based anomaly detection methods have been extensively studied across domains. Early approaches such as LSTM-based models and statistical thresholding methods demonstrated effectiveness in detecting temporal anomalies [32]. Recent work further explores multi-level attention mechanisms for modeling user behavior in real-time systems [33], as well as generative adversarial networks and autoencoder-based frameworks for anomaly detection in microservices [34]. Privacy and robustness considerations have also driven research into differential privacy mechanisms and secure model adaptation [35]. Additionally, uncertainty-aware anomaly detection frameworks improve robustness under noisy observations [36], while stochastic recurrent neural networks capture complex temporal dependencies in multivariate time series [37].

More recently, explainable and interpretable AI methods have been applied to system monitoring and health analysis, providing insights into model decisions and system behavior [38]. Structural regularization techniques have been introduced to improve generalization and reduce bias in large language model fine-tuning [39], while transfer learning methods enable adaptation of LLMs to low-resource scenarios [40]. Finally, unsupervised anomaly detection methods based on variational autoencoders and reconstruction-based techniques provide scalable solutions for detecting anomalies in large-scale web and cloud systems [41][42].

3. Model Design

At the methodological level, the overall framework uses the operational state sequence of a cloud computing system as the modeling object, uniformly representing multi-source monitoring data as a state vector that evolves. Let the observed state of the system at time t be:

$$x_t = [x_t^{(1)}, \ldots, x_t^{(d)}]^T$$

Each dimension corresponds to abstract features such as resource utilization, service latency, or dependency link status. To characterize the dynamic evolution of the system, a state transition function is introduced to model the dependencies between consecutive time steps, defined as follows:

$$h_t = f(h_{t-1}, x_t)$$

Here, h_t represents the system's operational representation in the latent space, used to compress historical states while retaining key evolutionary information. This representation provides a unified state basis for subsequent defect detection and counterfactual inference. This paper presents the overall model architecture, as shown in Figure 1.

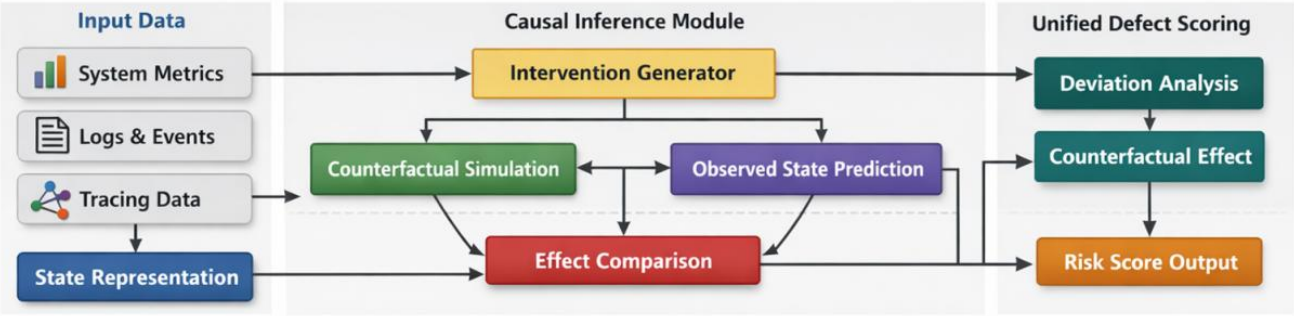

Figure 1. Overall model architecture

At the defect detection level, a state consistency function is constructed to characterize the system's behavior under normal operating conditions. Based on the hidden state h_t, the predicted state of the system is defined as:

$$x_{t+1} = g(h_t)$$

Potential anomalies are reflected by measuring the degree of deviation between the predicted state and the actual observation. The corresponding state deviation can be expressed as:

$$\Delta_t = \| x_{t+1} - \hat{x}_{t+1} \|_2$$

Where $\| \cdot \|_2$ represents the Euclidean norm. This deviation characterizes the inconsistency of the system over time, providing a quantitative basis for subsequent analysis of the causes of defects from a causal perspective.

To further enhance the ability to discern complex scenarios, a counterfactual modeling mechanism is introduced. This mechanism analyzes the impact of key factors by constructing

the system state after conditional intervention. Let z_t represent the set of interventionizable variables, such as resource allocation or scheduling strategies. Under a given intervention operation $do(z_t - z'_t)$, the counterfactual state is defined as:

$$x_{t+1}^{cf} = g(f(h_{t-1}, [x_t, z'_t]))$$

By comparing the difference between the true state and the counterfactual state, a measure of the counterfactual effect can be obtained:

$$\delta_t = \| x_{t+1} - x_{t+1}^{cf} \|_2$$

This quantity reflects the sensitivity of the system's behavior to changes in specific conditions, thus revealing potential key causal factors.

In the unified judgment phase, the original state deviation and counterfactual effects are integrated to construct a comprehensive defect scoring function, which measures the system's risk level at the current moment. This scoring function is defined as follows:

$$S_t = \alpha \Delta_t + (1 - \alpha)\delta_t$$

Here, $\alpha \in [0,1]$ is a tradeoff coefficient used to balance the roles of observational consistency and counterfactual sensitivity in defect detection. In this way, the method can not only identify abnormal changes in system operating status but also characterize the underlying mechanisms of defect generation from the perspective of conditional intervention, providing a more robust and interpretable basis for defect detection in cloud computing systems.

4. Implementation

4.1 Dataset

This study adopts the Google Cluster Workload Traces from 2019, which record Borg cluster workloads, as an open source dataset to support research on cloud computing system defect detection. The dataset is publicly released by Google and reflects real production scheduling and resource usage behavior in large-scale computing clusters. It covers operating state variations under multi-cluster, multi-task, and multi-tenant settings. It effectively represents the complex, chained relationships among resource contention, scheduling decisions, performance fluctuations, and anomaly propagation in cloud environments. These characteristics align well with the research focus on hidden defects, dynamic evolution, and cross-component coupling in cloud computing systems.

In terms of data content, the dataset records task-level and machine-level workload information and resource usage statistics at a time slice granularity. It includes job and task submission and scheduling records, resource requests and allocations, CPU usage distributions, and multiple attributes related to runtime behavior. Due to its large scale, long temporal span, and diverse scenarios, the dataset inherently exhibits non-stationarity and concept drift. It can simulate abnormal signals and defect-triggering conditions caused by policy changes, workload variations, and environmental disturbances in real cloud platforms. This provides sufficient temporal evidence and structural cues for building a unified modeling framework that links observed states, predictive deviations, counterfactual interventions, and risk scoring.

Regarding data utilization in this study, resource usage and scheduling states within each time window are aggregated into system observation vectors. These vectors form continuous state sequence inputs. At the same time, resource requests, resource reservations, and scheduling-related fields are treated as candidate intervenable factors. They are used to construct alternative execution trajectories under counterfactual condition changes. This design supports analysis of defect formation mechanisms through condition change and outcome difference relationships. Based on this dataset, large-scale cloud workload complexity beyond laboratory settings can be reproduced without relying on private operational logs. As a result, the research findings are more closely aligned with practical risk governance requirements in real cloud platforms.

4.2 Specific implementation

In the implementation process, the raw cloud workload trace data are first organized into a unified temporal format. Multi-dimensional indicators at the machine and task levels are aligned and aggregated within fixed time windows. This procedure produces continuous system observation sequences. To ensure comparability across features with different scales, numerical indicators are standardized or processed using robust normalization. Missing segments are handled through a combination of forward filling and within-window interpolation. This strategy reduces spurious anomalies caused by irregular sampling. In addition, state segments are constructed based on task lifecycles and scheduling event records. Key fields are mapped into a set of candidate intervenable variables, such as resource requests, resource limits, priorities, and scheduling constraints. These variables support condition replacement and consistency control in subsequent counterfactual generation.

For model construction, the system state representation module receives the observation vector at each time step and encodes it into latent state representations. This design captures temporal dependencies among resource contention, workload variation, and scheduling actions. A prediction branch then generates the next step state based on the latent representation. The resulting prediction is used to compute observation deviation and characterize operational inconsistency. The counterfactual branch employs an intervention generator to select or sample intervention schemes from the candidate variables. Under the assumption that other conditions remain unchanged, target variables are replaced to form counterfactual input sequences. A counterfactual simulation module applies the same state encoding mechanism to produce corresponding counterfactual evolution outcomes. To ensure the credibility of counterfactual construction, intervention magnitudes and valid ranges are constrained during implementation. For example, variable changes are restricted within historical quantile intervals. Intervention samples that violate scheduling rules are

directly removed. This process avoids generating unreasonable counterfactual trajectories.

At the output and deployment stage, observation deviations and counterfactual effects are integrated to produce a unified defect risk score. Risk curves and key intervention explanations are then reported at the time window level. A lightweight online inference workflow is adopted in practice. Data collection and feature aggregation are completed within a streaming pipeline. The model maintains only a sliding cache of recent windows, which ensures low latency and scalability. In parallel, a risk decomposition interface is provided. It returns the indicator dimensions and corresponding intervention variables that contribute most to the score. This design facilitates integration with alerting platforms and supports traceable defect localization. To adapt to different monitoring standards across cloud environments, the feature mapping layer uses configurable dictionaries and template-based aggregation rules. As a result, the same implementation can be transferred to different metric systems and sampling granularities without modifying the core model.

5. Experimental Results and Analysis

This paper first presents the experimental results compared with other models, as shown in Table 1.

Table 1: Comparative experimental results

Method	Acc	Precision	Recall	F1-Score
Deeplog[43]	0.8421	0.8174	0.8036	0.8104
Logbert[44]	0.8735	0.8542	0.8327	0.8433
SparseRCA[45]	0.8897	0.8671	0.8564	0.8617
RCAEval[46]	0.9024	0.8833	0.8719	0.8775
Microrca[47]	0.9186	0.9041	0.8927	0.8984
Ours	0.9478	0.9316	0.9264	0.9289

Overall comparison shows that the proposed method achieves more stable and more consistent advantages across multiple evaluation metrics. This indicates that, after integrating a counterfactual architecture, the model can not only capture abnormal patterns in cloud system operations but also more effectively distinguish true defect signals caused by resource fluctuations, scheduling disturbances, and dependency coupling. Compared with baseline methods that mainly rely on log sequence pattern learning, the proposed framework provides a more comprehensive characterization of cross-component propagation and temporal dependencies. As a result, defect identification is closer to real cloud environments, where defects are often hidden, gradually evolving, and influenced by multiple sources of interference.

From the perspective of accuracy, traditional log-based anomaly detection models are usually effective at identifying anomalies with stable templates or clear pattern shifts. However, when facing weak anomalies triggered by configuration drift, workload migration, or dependency changes, they tend to misclassify normal fluctuations as defects or miss early warning signals. The comparison results indicate that, after introducing a counterfactual condition change perspective, the model can suppress interference from incidental noise through sensitivity assessment of key conditions. This leads to more reliable overall judgments. Such capability is particularly suitable for cloud operating scenarios with continuous distribution changes caused by multi-tenant sharing and elastic scaling.

Considering the coordination between precision and recall, many existing methods face a clear trade-off between reducing false alarms and minimizing missed detections. Improving recall often increases false positives, while improving precision may sacrifice the ability to capture latent defects. The proposed method maintains both high precision and broad coverage in the comparison. This suggests that the model can identify abnormal evidence that is truly related to defects in complex environments, while still tracking critical events when defect signals are weak or propagation chains are long. This balance is especially important for cloud defect detection, since false alarms increase operational burden, whereas missed detections can allow defects to spread and amplify business risk.

With respect to composite metrics, the results demonstrate more pronounced improvements in overall stability and robustness. Compared with detection approaches based mainly on correlation matching, the counterfactual architecture emphasizes result consistency and difference under condition changes. This design strengthens focus on key causal factors at the mechanism level. It therefore enables more consistent metric improvements under typical cloud scenarios such as environmental variation, workload switching, and dependency reconfiguration.

The learning rate determines the step size and directional stability of parameter updates, and is a key factor affecting the model's training dynamics and convergence behavior. Different learning rates alter how the model adapts to noise perturbations and gradient scaling, thus affecting the quality of feature representation formation. To evaluate the robustness of the method under changes in training configuration, it is necessary to systematically perturb the learning rate and observe the sensitivity of performance to these changes. The experimental results are shown in Figure 2.

From the learning rate sensitivity curves, it can be observed that the proposed method exhibits a clear nonlinear response to the choice of optimization step size. This indicates that, in high noise temporal scenarios such as cloud computing system defect detection, the learning rate not only affects convergence speed but also alters the balance between short-term disturbances and long-term dependencies. When the learning rate falls within a moderate range, the model can more stably absorb informative signals from multiple source monitoring data. This leads to more consistent state representations and facilitates the identification of abnormal patterns induced by resource contention and dependency chain coupling during defect evolution.

Regarding overall correctness-related performance, the curves show a noticeable decline once the learning rate deviates from a reasonable range. This reflects that an excessively large update step may cause parameter oscillation under complex non-stationary distributions. As a result, the model struggles to maintain a robust representation of normal behavior patterns, which leads to unstable decision boundaries in defect detection. In cloud environments, workload

fluctuations, elastic scaling, and scheduling reallocation continuously change system state distributions. If the learning rate is too high, the model becomes more susceptible to transient spikes or incidental jitter. This effect can cause true defect signals to be obscured by noise.

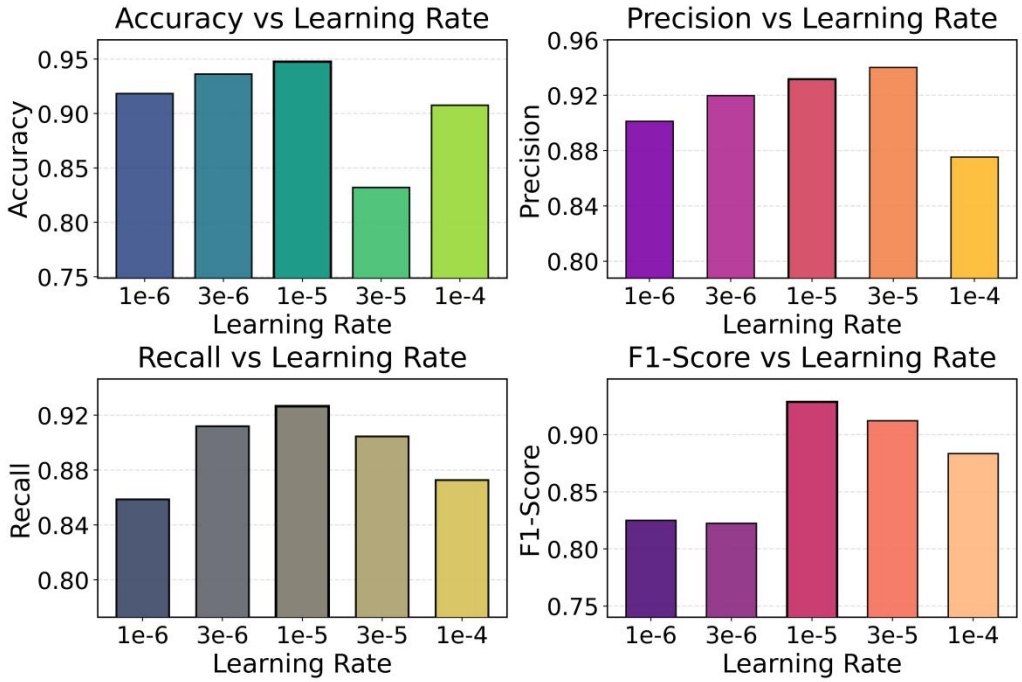

Figure 2. The impact of the learning rate on experimental results

From the perspective of false alarm control and positive class coverage, the trends across different subplots are not fully consistent. This suggests that the learning rate influences the model's selective focus on defect evidence in different ways. On one hand, a more aggressive learning rate may temporarily enhance responsiveness to prominent anomalies on certain metrics. At the same time, it can weaken the ability to track fine-grained and gradually evolving defect signals. This reduction affects coverage of defects with long propagation chains. On the other hand, a more conservative learning rate helps suppress misjudgments caused by fluctuations. However, it may also lead to insufficient representation of weak defects or early warning signals, which manifests as reduced detection sensitivity. These differences indicate that, in cloud defect detection tasks, the learning rate is not only a parameter for training stability but also directly shapes preferences in defect evidence aggregation and decision making.

Changes in composite metrics further confirm that the proposed framework has a relatively clear optimal operating range for the learning rate. Within this range, different performance aspects become more coordinated. This indicates that the model can form more stable risk scores when integrating state prediction deviations and counterfactual effects. For methods that incorporate counterfactual architectures, an excessively large learning rate may disrupt alignment between the counterfactual branch and the observation branch. This disruption can cause drift in intervention sensitivity estimation and weaken the reliability of mechanism-level explanations. Conversely, an excessively small learning rate may prevent both branches from adequately learning complex dependency structures. Overall, this sensitivity analysis highlights the important role of a properly chosen learning rate in improving stability and interpretability consistency for cloud computing system defect detection.

The length of the time window determines the historical range over which the model aggregates system state information, directly affecting how short-term jitter and long-term dependencies are characterized. For defect detection in cloud computing systems, different window lengths alter the distinguishability and temporal consistency of defect symptoms; therefore, sensitivity analysis is needed to verify the robustness of the method to the window setting. The experimental results are shown in Figure 3.

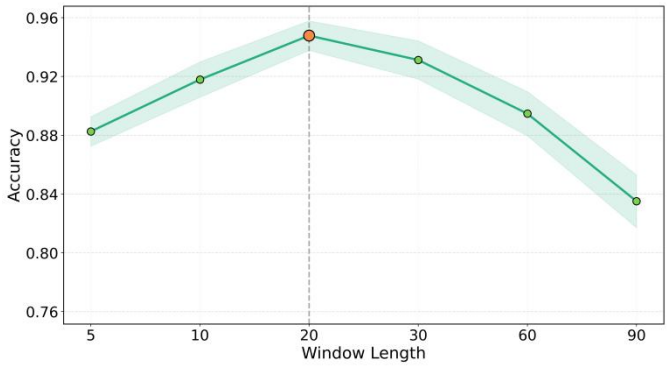

Figure 3. Experiment on the sensitivity of the time window length to accuracy

From the overall trend of varying time window lengths, model performance shows a clear dependence on the state aggregation scale. This observation indicates that defect

formation in cloud computing systems exhibits multi-scale characteristics along the temporal dimension. When the time window is too short, it fails to fully capture the evolution of resource contention and service dependencies. The model then focuses more on instantaneous fluctuations and overlooks potential cumulative risks. In contrast, when the window is too long, system states become overly smoothed. Critical abnormal signals are easily masked by normal behavior, which weakens the ability to identify defect indicators.

Around a moderate window length, the curves display a more stable and concentrated advantage region. This suggests that such a scale achieves a better balance between short-term fluctuations and long-term dependencies. This behavior is consistent with the actual evolution of defects in cloud environments. Most defects are not triggered instantaneously. They gradually accumulate within a certain time range due to configuration drift, workload variation, or scheduling adjustments. An appropriate window length helps the model capture this progressive change and produces state representations that better align with real system dynamics.

As the time window continues to increase, accuracy shows a clear decline. This indicates that an excessively wide historical range introduces a large amount of information that is irrelevant to the current decision. It increases redundancy and noise in the state representation. In cloud platforms, workload patterns and scheduling policies can differ significantly across time periods. When overly long histories are aggregated together, the model becomes vulnerable to interference from outdated behavior patterns. This reduces sensitivity to current abnormal situations. This phenomenon highlights the critical role of temporal context selection in dynamic cloud environments.

Taken together, this sensitivity analysis demonstrates that the proposed method exhibits a clear and interpretable response to time window configuration. A well-chosen window length not only improves the stability of state modeling but also helps the counterfactual branch maintain consistency with the actual system evolution rhythm when constructing condition changes. This consistency enhances the reliability of defect risk scoring. These results further indicate that, in cloud computing system defect detection frameworks that integrate counterfactual architectures, time scale selection is one of the key factors influencing model performance and mechanism consistency.

The noise injection ratio is used to simulate jitter, missing data, and abnormal disturbances that occur during the acquisition and transmission of cloud computing monitoring data, thereby verifying the stability of the model under non-ideal observation conditions. By changing the noise intensity, the robust characterization ability of the proposed method to input disturbances and its preservation of defective signals can be verified. The experimental results are shown in Figure 4.

From the trend of varying noise injection ratios, it can be observed that the discriminative capability of the model gradually decreases as stronger perturbations are applied to the observations. This behavior is consistent with the characteristics of cloud monitoring data under non-ideal acquisition and transmission conditions. Noise directly disrupts the original temporal consistency of metrics and log sequences. It makes defect signals harder to separate from normal fluctuations and imposes higher robustness requirements on defect detection. This trend indicates that data quality itself is a critical factor affecting the reliability of defect identification in cloud environments.

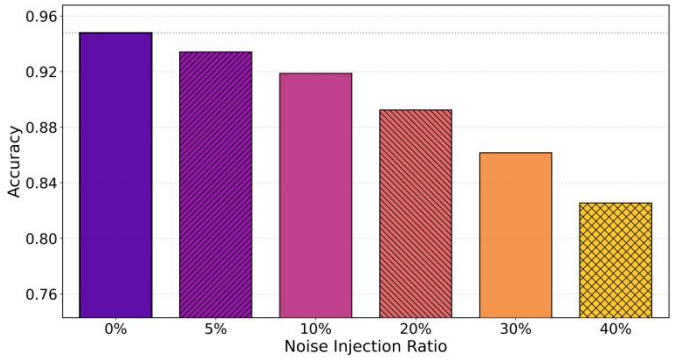

Figure 4. Sensitivity experiment of noise injection ratio to accuracy

Under low to moderate noise levels, the performance degradation remains relatively smooth. This suggests that the proposed method has a certain tolerance to minor jitter and localized abnormal sampling. Such stability aligns with the unified state representation and consistency modeling adopted in the method. The model can retain aggregation of dominant operational patterns even when short-term disturbances exist. It is not easily dominated by isolated spikes or incidental missing values. For cloud defect detection based on multi-source data fusion, this property implies that reliable risk assessment can still be maintained under common monitoring noise conditions.

When noise intensity increases further, the performance decline becomes more pronounced. This reflects that strong perturbations can systematically damage the separability of defect cues. Higher noise ratios increase randomness at the feature level and may obscure or incorrectly amplify abnormal segments along key dependency chains. As a result, the model finds it more difficult to form stable temporal alignment. For frameworks that integrate counterfactual architectures, strong noise also interferes with intervention sensitivity estimation. The response of the counterfactual branch to condition changes becomes diluted by noise components, which weakens mechanism-level decision support.

6. Conclusion

This work focuses on key challenges in cloud computing systems, including difficult defect localization, hidden evolution processes, and complex propagation paths. It systematically investigates a defect detection method that integrates a counterfactual architecture. By modeling multi-source operational data as unified temporal state representations and introducing conditional interventions and counterfactual inference, the proposed approach overcomes the limitations of traditional correlation-driven detection in complex cloud environments. Defect identification is no longer restricted to surface-level anomalies. Instead, it incorporates underlying mechanisms to support more discriminative judgments. Overall, the study provides a more structured

analytical perspective for understanding defect formation and evolution in cloud computing systems.

From a methodological perspective, the proposed unified framework establishes an effective connection between state prediction deviation and counterfactual effects. This design enables the defect detection process to consider both observational consistency and sensitivity to condition changes. It not only improves the stability of defect discrimination but also offers interpretable support for analyzing multi-factor coupling in complex systems. Compared with detection approaches that rely solely on historical pattern matching, the counterfactual architecture emphasizes whether system behavior changes fundamentally under hypothetical condition variations. This focus makes it easier to identify key factors that truly determine defect occurrence. Such a perspective has important practical implications for cloud platforms characterized by high resource sharing and frequent scheduling changes.

At the application level, the findings of this study provide direct reference value for cloud platform operations management, business continuity assurance, and risk early warning. Defect detection mechanisms with counterfactual analysis capability can offer clearer risk indications for operators. This reduces the handling burden caused by false alarms and lowers the likelihood that hidden defects escalate into system-level failures. The framework does not rely on private rules of specific cloud vendors. It therefore exhibits strong generality and transferability. It can adapt to cloud computing systems of different scales and architectures, and provides intelligent support for the stable operation of large-scale distributed platforms.

Looking ahead, defect detection research that incorporates counterfactual inference still has broad development potential. On the one hand, deeper integration with automated operations decision-making can be explored. Counterfactual analysis results could directly inform scheduling optimization and resource adjustment strategies. This would promote a shift from reactive response to proactive prevention in cloud systems. On the other hand, this idea can be extended to more complex scenarios such as edge computing, industrial internet, and critical infrastructure. It can offer a unified modeling paradigm for risk governance in high-reliability systems. Overall, this work lays a foundation for building intelligent cloud operations frameworks with a mechanism understanding capability and contributes positively to research and applications in related fields.

References

[1] Markakis, M., Youngmann, B., Gao, T., et al., "From logs to causal inference: Diagnosing large systems," Proceedings of the VLDB Endowment, vol. 18, no. 2, pp. 158-172, 2024.

[2] Ikram, A., Chakraborty, S., Mitra, S., et al., "Root cause analysis of failures in microservices through causal discovery," Advances in Neural Information Processing Systems, vol. 35, pp. 31158-31170, 2022.

[3] Li, M., Li, Z., Yin, K., et al., "Causal inference-based root cause analysis for online service systems with intervention recognition," Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 3230-3240, 2022.

[4] Sulem, D., Donini, M., Zafar, M. B., et al., "Diverse counterfactual explanations for anomaly detection in time series," arXiv preprint arXiv:2203.11103, 2022.

[5] Zhu, Y., Wang, J., Li, B., et al., "Root cause localization for microservice systems in cloud-edge collaborative environments," arXiv preprint arXiv:2406.13604, 2024.

[6] Li, Z., Chen, J., Jiao, R., et al., "Practical root cause localization for microservice systems via trace analysis," Proceedings of the 2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS), pp. 1-10, 2021.

[7] Zhang, C., "Reinforcement Learning-Based Dynamic Cache Replacement Using Deep Q-Networks," Transactions on Computational and Scientific Methods, vol. 4, no. 12, 2024.

[8] Pearl, J., Causal Inference in Statistics: An Overview, 2009.

[9] Wang, Y., "Semantic-Driven Large Model Scheduling for Distributed Systems via Unified Representation and Policy Generation," Transactions on Computational and Scientific Methods, vol. 4, no. 12, 2024.

[10] Bottou, L., Peters, J., Quiñonero-Candela, J., Charles, D. X., Chickering, D. M., Portugaly, E., and Snelson, E., "Counterfactual reasoning and learning systems: The example of computational advertising," Journal of Machine Learning Research, vol. 14, no. 11, 2013.

[11] Zhang, Q., "Adaptive Resource Scheduling in Distributed Computing via Multi-Agent Reinforcement Learning and Graph Convolutional Modeling," Transactions on Computational and Scientific Methods, vol. 4, no. 11, 2024.

[12] Hu, Y., "Autonomous Agent Architecture for Complex Tasks via Hierarchical Planning and Language Model Reasoning," Transactions on Computational and Scientific Methods, vol. 4, no. 12, 2024.

[13] Reiss, C., Wilkes, J., and Hellerstein, J. L., "Google cluster-usage traces: format + schema," Google Inc., White Paper, vol. 1, pp. 1-14, 2011.

[14] Karimi, A. H., Schölkopf, B., and Valera, I., "Algorithmic recourse: from counterfactual explanations to interventions," Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, pp. 353-362, 2021.

[15] Gan, Q., "Large Language Model Framework for Multi-Document Financial Anomaly Detection in Intelligent Auditing via Semantic Mapping and Risk Reasoning," Transactions on Computational and Scientific Methods, vol. 4, no. 12, 2024.

[16] Chen, F., "AI-Augmented Anomaly Detection via Generative Distribution Modeling and Uncertainty Quantification in Cloud Systems," Transactions on Computational and Scientific Methods, vol. 4, no. 11, 2024.

[17] Grandl, R., Ananthanarayanan, G., Kandula, S., Rao, S., and Akella, A., "Multi-resource packing for cluster schedulers," ACM SIGCOMM Computer Communication Review, vol. 44, no. 4, pp. 455-466, 2014.

[18] Islam, M. S., and Manik, M. M. T. G., "Explainable AI in Healthcare: Leveraging Machine Learning and Knowledge Representation for Personalized Treatment Recommendations," Journal of Posthumanism, vol. 5, no. 1, pp. 1541-1559.

[19] Hua, C., "A Semantic-Prior-Guided AI Framework for Collaborative Environment Understanding and Robust Agent Decision Making," Transactions on Computational and Scientific Methods, vol. 4, no. 12, 2024.

[20] Tan, R., and Li, Z., "Maad: A distributed anomaly detection architecture for microservices systems," Proceedings of the 2024 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 1009-1021, 2024.

[21] Chiang, C., "Drift-Aware Adaptive Classification for Imbalanced Data via Dynamic Class Reweighting and Structural Regularization," Transactions on Computational and Scientific Methods, vol. 4, no. 12, 2024.

[22] Nie, C., "Representation Learning with Multi-Task Self-Supervision for Structurally Diverse Spatiotemporal Time Series Forecasting," Journal of Computer Technology and Software, vol. 3, no. 7, 2024.

[23] He, S., Zhu, J., He, P., and Lyu, M. R., "Experience report: System log analysis for anomaly detection," Proceedings of the 2016 IEEE 27th

International Symposium on Software Reliability Engineering (ISSRE), pp. 207-218, 2016.

[24] Wang, T., and Qi, G., "A comprehensive survey on root cause analysis in (micro) services: Methodologies, challenges, and trends," arXiv preprint arXiv:2408.00803, 2024.

[25] Lai, J., "Attention Alignment under Logical Constraints for Reliable Financial Statement Reasoning," Transactions on Computational and Scientific Methods, vol. 4, no. 12, 2024.

[26] Nobre, J., Pires, E. S., and Reis, A., "Anomaly detection in microservice-based systems," Applied Sciences, vol. 13, no. 13, 2023.

[27] Le, V. H., and Zhang, H., "Log-based anomaly detection without log parsing," Proceedings of the 2021 IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 492-504, 2021.

[28] Li, J., "LocateNet: Large Multimodal Models for Text-Guided Object Localization," Transactions on Computational and Scientific Methods, vol. 4, no. 12, 2024.

[29] Xing, Y., "Enhancing Advertising Recommendation Performance via Integrated Causal Inference and Exposure Bias Correction," Journal of Computer Technology and Software, vol. 2, no. 3, 2023.

[30] Zhu, L., Cui, W., Xing, Y., and Wang, Y., "Collaborative optimization in federated recommendation: Integrating user interests and differential privacy," Journal of Computer Technology and Software, vol. 3, no. 8, 2024.

[31] Wang, J., "Multivariate time series forecasting and classification via GNN and transformer models," Journal of Computer Technology and Software, vol. 3, no. 9, 2024.

[32] Hundman, K., Constantinou, V., Laporte, C., Colwell, I., and Soderstrom, T., "Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding," Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 387-395, 2018.

[33] Wang, M., "Multi-Level Attention and Sequence Modeling for Dynamic User Interest Representation in Real-Time Advertising Recommendation," Transactions on Computational and Scientific Methods, vol. 3, no. 2, 2023.

[34] Ma, Y., "Anomaly detection in microservice environments via conditional multiscale GANs and adaptive temporal autoencoders," Transactions on Computational and Scientific Methods, vol. 4, no. 10, 2024.

[35] Li, Y., "Task-aware differential privacy and modular structural perturbation for secure fine-tuning of large language models," Transactions on Computational and Scientific Methods, vol. 4, no. 7, 2024.

[36] Qiu, Z., "A Multi-Scale Deep Learning and Uncertainty Estimation Framework for Comprehensive Anomaly Detection in Cloud Environments," Transactions on Computational and Scientific Methods, vol. 3, no. 2, 2023.

[37] Su, Y., Zhao, Y., Niu, C., Liu, R., Sun, W., and Pei, D., "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2828-2837, 2019.

[38] Sun, X., Yao, Y., Wang, X., Li, P., and Li, X., "AI-driven health monitoring of distributed computing architecture: Insights from XGBoost and SHAP," Proceedings of the 2024 4th International Conference on Communication Technology and Information Technology (ICCTIT), pp. 480-484, 2024.

[39] Liu, H., "Structural regularization and bias mitigation in low-rank fine-tuning of LLMs," Transactions on Computational and Scientific Methods, vol. 3, no. 2, 2023.

[40] Deng, Y., "Transfer methods for large language models in low-resource text generation tasks," Journal of Computer Science and Software Applications, vol. 4, no. 6, 2024.

[41] Xu, H., Chen, W., Zhao, N., Li, Z., Bu, J., Li, Z., and Qiao, H., "Unsupervised anomaly detection via variational auto-encoder for seasonal KPIs in web applications," Proceedings of the 2018 World Wide Web Conference, pp. 187-196, 2018.

[42] Audibert, J., Michiardi, P., Guyard, F., Marti, S., and Zuluaga, M. A., "USAD: Unsupervised anomaly detection on multivariate time series," Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 3395-3404, 2020.

[43] Du, M., Li, F., Zheng, G., et al., "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 1285-1298, 2017.

[44] Guo, H., Yuan, S., and Wu, X., "LogBERT: Log anomaly detection via BERT," Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN), pp. 1-8, 2021.

[45] Yao, Z., Ye, H., Pei, C., et al., "SparseRCA: Unsupervised root cause analysis in sparse microservice testing traces," Proceedings of the 2024 IEEE 35th International Symposium on Software Reliability Engineering (ISSRE), pp. 391-402, 2024.

[46] A. Ikram, S. Chakraborty, S. Mitra, S. Saini, S. Bagchi and M. Kocaoglu, "Root cause analysis of failures in microservices through causal discovery," Advances in Neural Information Processing Systems, vol. 35, pp. 31158-31170, 2022.

[47] Wu, L., Tordsson, J., Elmroth, E., et al., "MicroRCA: Root cause localization of performance issues in microservices," Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS), 2020.