Transactions on Computational and Scientific Methods | Vo. 4, No. 1, 2024
ISSN: 2998-8780
https://pspress.org/index.php/tcsm
Pinnacle Science Press

Behavior Representation Learning for Reliable Monitoring of AI Inference Backends

Tian Ding[1], Ye Yuan[2]
[1]Boston University, Boston, USA
[2]University of Southern California, Los Angeles, USA
*Corresponding Author: Ye Yuan; yuanyuan.1189@gmail.com

Abstract: In multi-tenant and multi-model concurrent AI inference backends, system behavior exhibits high dynamics and complex structure. This property poses significant challenges to monitoring mechanisms based on rules or single metrics. From the perspective of system behavior modeling, this paper proposes a learning driven backend monitoring approach. Multi-source observations generated during inference execution are mapped into a unified latent representation space. This mapping captures the global evolution of system states. By jointly modeling temporal consistency and global consistency of behavior representations, the method forms stable and discriminative behavior structures in the latent space. These structures provide a reliable basis for runtime state assessment. On this foundation, a unified monitoring scoring mechanism is constructed. It quantifies the deviation between the current system state and the overall behavior pattern. This design enables effective identification of anomalous behavior. Comparative analysis with multiple monitoring methods shows that system behavior learning based monitoring achieves clear advantages in discrimination stability and false alarm control. The method adapts well to complex operating conditions in inference backends. This work presents a system-level modeling perspective for reliable AI inference backend management. It supports the practical adoption of learning based monitoring methods in real-world backend systems.

Keywords: System behavior modeling; Backend monitoring; Unified representation learning; Anomaly discrimination

1. Introduction

As artificial intelligence models are widely deployed in online services, backend inference systems become a core infrastructure for the stable operation of intelligent applications. From recommendation and search ranking to multimodal content generation, an increasing amount of business logic relies on continuous and high-concurrency model inference. Compared with traditional backend systems, AI inference backends exhibit stronger dynamics and higher complexity in their runtime behavior[1]. Execution is affected by request arrival patterns and scheduling policies, as well as by model architectures, computation graphs, and internal inference states. This multi-factor coupling leads to highly nonlinear and non-stationary behavior, which introduces new challenges for system stability and reliable monitoring[2].

In real-world environments, AI inference backends usually operate under multi-tenant and multi-model concurrency. Service instances share computation, storage, and communication resources. Resource contention, workload fluctuation, and model heterogeneity jointly drive continuous changes in system states over time and space. Under such conditions, anomalies rarely appear as obvious deviations in a single metric.

They are often reflected in coordinated changes across multiple behavioral dimensions. For example, structural imbalance may arise among inference latency, resource utilization, and scheduling behavior, while each metric remains within an acceptable range. If such anomalies are not detected in time, risks accumulate during long-term operation and may eventually degrade service quality or trigger system-level failures. Effective modeling and monitoring of inference backend behavior, therefore, becomes essential for system reliability.

Conventional backend monitoring and anomaly detection methods mainly rely on predefined rules or statistical assumptions. Key metrics are selected based on human experience, and fixed thresholds are applied to judge system states. These approaches are useful when system structures are stable and workload patterns are predictable. However, they are often ineffective for complex AI inference backends. Fixed rules cannot adapt to rapidly changing inference workloads or evolving model behavior. Monitoring based on isolated metrics also fails to capture interactions among system components, which leads to missed structural anomalies. As system scale and operational diversity increase, monitoring mechanisms that depend on strong prior assumptions show limited adaptability and higher risks of misjudgment.

Learning driven approaches provide new opportunities for modeling complex system behavior. By learning from historical runtime data, system behavior patterns can be captured without explicitly defined rules. In AI inference backends, inference requests, resource allocation, and execution feedback form time series with inherent consistency and correlation[3]. This property offers a natural foundation for behavior learning. Unlike traditional methods that focus on local metric variations, behavior-based monitoring has greater potential to understand system states at a global level. It enables the identification of coordination among components and the evolution of system dynamics, which supports more discriminative anomaly detection and runtime assessment.

From the perspective of system behavior modeling, developing a learning driven monitoring framework for AI inference backends is therefore of significant importance. Such a framework supports the transition from static rule-based monitoring to adaptive learning based mechanisms[4]. It allows the system to maintain continuous awareness of its own state under complex operating conditions. Moreover, structured representations of inference backend behavior provide a more robust basis for reliability assessment. A behavior-centric monitoring paradigm contributes to improved stability and controllability of long-running AI inference services. It also opens new research directions for the operation and management of intelligent infrastructure.

2. Overview of Existing Work

Early studies on backend system monitoring and anomaly detection mainly focused on rule-driven and statistical modeling approaches. These methods are usually based on prior understanding of system behavior. Key performance metrics are manually selected, and thresholds or statistical distributions are defined to judge whether the system deviates from normal states. Such approaches achieved practical results in traditional service architectures and relatively stable workload environments. They are especially interpretable when system behavior changes are limited, and anomaly patterns are clear. However, these methods rely on strong assumptions, such as stationarity of system state distributions or independence among metrics. Once operating conditions change significantly, monitoring rules require frequent adjustment, which limits their long-term adaptability to complex system evolution[5].

As system scale increases and runtime environments become more dynamic, some studies introduce data-driven anomaly detection methods. These approaches learn normal behavior patterns from historical runtime data. Feature modeling, distance measurement, and probabilistic inference are commonly used to represent multidimensional monitoring data. This reduces reliance on manually defined rules. To some extent, learning based methods improve robustness to workload variation and noise. However, most existing studies focus on metric-level modeling. They emphasize numerical distributions or time series characteristics[6]. Explicit modeling of structural relationships among system components and execution stages is often missing. As a

result, modeling capacity remains limited in complex inference pipelines and multi-model collaborative scenarios.

For monitoring AI inference systems, recent work also investigates performance analysis and anomaly identification during model execution. These studies usually focus on metrics such as inference latency, throughput, and resource consumption. The goal is to assess system status through key statistical properties of the inference process. However, AI inference backends differ from traditional services. Runtime behavior is influenced not only by scheduling and resource allocation, but also by model architecture, input data characteristics, and inference execution paths. In multi-tenant and multi-model concurrent settings, interactions among inference tasks further increase behavioral complexity. Monitoring methods based on a single performance perspective, therefore, struggle to reflect overall system behavior[7].

Recent studies further indicate that backend reliability requires coupling monitoring with adaptive control and representation-level anomaly modeling. Reinforcement learning has been used for container elasticity under changing workload intensity [8], while cloud-edge collaborative inference partitions DNN execution across heterogeneous resources to balance latency, throughput, and energy constraints [9]. Semantic log representations improve robustness to unstable system events and changing log templates [10], and learning-based cluster scheduling demonstrates how learned policies can guide large-scale distributed runtime decisions under stochastic arrivals [11]. Production time-series anomaly detection services further show that online monitoring pipelines need data ingestion, real-time scoring, and experimentation support to remain effective in operational settings [12]. Transformer-based multivariate anomaly detection improves temporal context modeling for complex operational signals [13]. Taken together, these studies show that backend monitoring should not be limited to isolated thresholds or single metrics. Instead, monitoring logic needs to combine behavior representation, execution context, adaptive control, and time-series anomaly scoring, which is consistent with the system behavior learning perspective adopted in this paper.

In summary, existing research provides a solid foundation for backend monitoring and anomaly detection. However, unified modeling at the system behavior level remains insufficient. Most methods fail to capture the coordinated evolution of multi-source information in AI inference backends from a global perspective. This limits the ability to reveal underlying behavioral structure. In addition, behavior consistency and drift patterns during long-term inference operation are still underexplored. From a system behavior learning perspective, structured modeling of AI inference backends offers a promising direction. It can bridge the gap between metric-level monitoring and system-level understanding. This perspective supports the development of more reliable and scalable backend monitoring mechanisms.

3. Framework Design

3.1 Overall Framework Description

This paper treats the AI inference backend as a continuously running dynamic system, exhibiting evolving behavior over time as it processes inference requests. The overall model architecture is shown in Figure 1.

At each time step, the system exposes a set of observable signals to characterize operational features such as request processing status, resource usage, and scheduling decision results. This multi-source information is uniformly modeled as a snapshot of system behavior, serving as the foundational representation for backend monitoring. Formally, the system behavior at time step t is represented as a multi-dimensional observation vector:

$$x_t = [x_t^{(1)}, x_t^{(2)}, \ldots, x_t^{(d)}]$$

Different dimensions correspond to different operational attributes of the inference backend at that time step. As the system continues to run, a system behavior sequence $\{x_t\}_{t=1}^{T}$ is obtained, which characterizes the overall evolution of the backend execution state. The goal of the method in this paper is to learn a stable and compact representation of system behavior from this behavior sequence without relying on manual rules or fixed thresholds, providing a unified modeling foundation for backend operation monitoring.

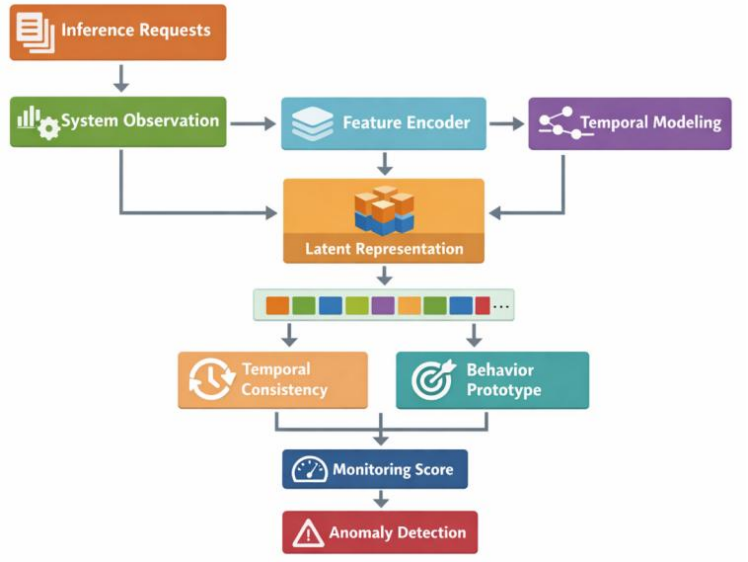

Figure 1. Overview of the proposed method.

3.2 System Behavior Representation Learning

To characterize the intrinsic relationships between different attributes in system behavior, this paper maps the original observation vectors to a latent behavior space to obtain a structured representation of system behavior. This mapping is implemented through a learnable function to compress high-dimensional observations and highlight key operational modes. The latent behavior representation at time step t is defined as:

$$z_t = f_\theta(x_t)$$

Here, $f_\theta(\bullet)$ represents the parameterized behavior mapping function. To ensure the stability of the latent behavior representation in the time dimension, a behavior consistency constraint between adjacent time steps is introduced to suppress irrelevant fluctuations caused by short-term perturbations. This constraint is achieved by minimizing the differences between consecutive latent representations:

$$L_{temp} = \sum_{t=2}^{T} \| z_t - z_{t-1} \|_2^2$$

3.3 Behavioral Consistency Modeling

Relying solely on local temporal consistency is insufficient to characterize the overall behavioral features of a system over a longer operating period. Therefore, this paper further constructs a global behavioral reference in the latent space to describe the overall behavioral patterns of the system under stable operating conditions. This reference behavior is obtained by time-averaging the latent representation:

$$\bar{z} = \frac{1}{T} \sum_{t=1}^{T} z_t$$

Based on this, the deviation between the potential behavior representation at each time step and the global reference behavior is regarded as the basis for characterizing the degree of behavioral consistency, and corresponding consistency constraints are defined:

$$L_{cons} = \sum_{t=1}^{T} //z_t - \bar{z}//_2^2$$

By simultaneously modeling local temporal continuity and global behavioral consistency, the latent behavioral space can take into account both short-term dynamic changes and long-term operational patterns, providing a stable representation for understanding system-level behavior.

D. Monitoring Score Construction

Based on the learned system behavior representation, this paper further constructs a unified monitoring score to quantify the deviation between the current system state and its overall behavior pattern. For any time step t , the system behavior offset score is defined as:

$$s_t = //z_t - \bar{z}//_2$$

This score reflects the degree of change in the system's behavioral state at that moment relative to the global reference behavior. To obtain a smoother and more stable monitoring signal, the scores at single moments are aggregated over time to obtain the overall system state indication:

$$S = \frac{1}{T} \sum_{t=1}^{T} s_t$$

The aforementioned monitoring and scoring method is based on the learned representation of system behavior, providing a unified and scalable modeling foundation for the continuous perception and reliable monitoring of the AI inference backend's operating status.

4. Experimental Analysis

4.1 Dataset

This study uses the Alibaba Cluster Trace Dataset as the experimental data source. The dataset is collected from a real large-scale data center operating environment and records job scheduling, resource allocation, and system load variations during continuous cluster operation. The data are organized as a time series and cover multiple computing nodes and a large number of concurrent tasks. They reflect dynamic behavior characteristics of complex backend systems under real operating conditions. Since the data originate from production environments, the operating patterns exhibit strong heterogeneity and non-stationarity. This property makes the dataset suitable for modeling system behavior evolution over time.

The dataset contains diverse observations closely related to backend operating states, including task arrival and execution processes, resource utilization, and scheduling-related events. These multi-source signals form continuous system behavior trajectories along the temporal dimension. They support system-level modeling of backend operation processes. By organizing observation signals at different time steps in a unified manner, behavior sequences that represent overall system states can be constructed. This process provides fundamental inputs for subsequent behavior representation learning and monitoring modeling. The dataset design avoids excessive reliance on single metrics and enables characterization of system behavior from a multi-dimensional perspective.

As a publicly available dataset, the Alibaba Cluster Trace Dataset offers strong reproducibility and broad research applicability. It has been widely used in studies on distributed system analysis and resource management. Its data scale and temporal span cover diverse operating state variations, which gives it strong representativeness for research on backend system behavior modeling, operational monitoring, and anomaly identification. Researching this dataset helps validate the applicability of the proposed method in real

complex system environments. It also provides realistic data support for monitoring studies targeting practical AI inference backends.

4.2 Experimental Results

This article first presents the results of the comparative experiments, as shown in Table 1.

Table 1: Comparative experimental results

Method	Precision	Recall	F1-score	False Positive Rate
DeepLog [14]	0.81	0.76	0.78	0.14
LogAnomaly [15]	0.83	0.79	0.81	0.12
OmniAnomaly [16]	0.85	0.80	0.82	0.11
MTAD-GAT [17]	0.86	0.82	0.84	0.10
Anomaly Transformer [18]	0.87	0.83	0.85	0.10
Ours	0.90	0.87	0.88	0.07

Comparative analysis across multiple monitoring approaches shows that modeling based on system behavior representations delivers more stable performance across several dimensions of anomaly discrimination. Methods that rely on unified behavior patterns capture runtime dynamics more comprehensively than approaches based on isolated features or local metrics. As a result, even under complex conditions with workload fluctuation or resource contention, the method preserves an accurate description of system states. This leads to a more robust trend in detection accuracy.

From the perspective of anomaly capture capability, conventional methods often fail to identify fine-grained state deviations promptly. This limitation mainly arises from their reliance on local statistical features. Behavior representation learning jointly models temporal sequences and structural relationships. State variations are therefore amplified in the latent space. This property provides an advantage in anomaly detection. Comparisons with multiple methods indicate that unified behavior modeling is more sensitive to state shifts and achieves higher completeness in anomaly discovery.

In terms of overall discrimination performance, monitoring mechanisms based on behavior consistency achieve a more balanced trade-off between precision and recall. Traditional approaches may exhibit bias in certain scenarios due to excessive dependence on single signals, which results in unstable decisions. By strengthening structural consistency and temporal consistency in representations, behavior modeling frameworks maintain coherence of monitoring signals across different execution phases. The resulting decisions are more stable and self-consistent. This comprehensive behavior indicates that system-level structured modeling is more effective in preserving reliable discrimination under dynamic conditions.

Clear differences are also observed in reducing false alarm risk. Because behavior modeling constructs a relatively stable reference pattern in the latent space, monitoring scores remain smooth when the system operates within normal regimes. Sensitivity to transient noise or local disturbances is reduced. Consequently, the overall false positive rate remains low. This helps avoid excessive alarm triggering and aligns monitoring outcomes with real operating conditions. Such a property is particularly important for long-running inference backends and provides a more reliable basis for runtime state assessment.

The unified representation dimension characterizes the expressive capacity of system behavior in the latent space. Its value directly determines the degree of information compression and the preservation of structural integrity. Different dimensional settings modify the distribution of system states in the representation space. This variation affects the stability of subsequent monitoring and decision processes. To analyze the influence of this factor on system monitoring capability, it is necessary to examine how relevant performance evolves under multiple representation dimension configurations, and the experimental results are shown in Figure 2.

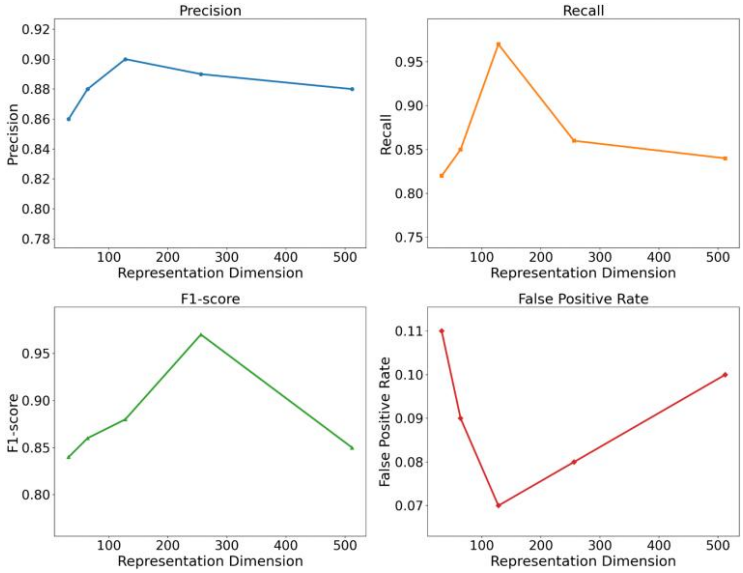

Figure 2. The impact of changes in a unified representation dimension on the system's ability to discriminate behavior and the stability of monitoring.

Under different representation dimension settings, system monitoring performance exhibits clear nonlinear variation patterns. This behavior reflects the direct influence of unified representation capacity on system behavior modeling. At lower dimensions, the latent space has limited ability to characterize runtime information. It fails to fully represent the multidimensional behavior structure of complex inference backends. As the representation dimension increases, the distribution of system behavior in the latent space becomes more distinguishable. This allows the monitoring mechanism to capture critical changes in runtime states more effectively. These observations indicate that unified representations play an important role in balancing information compression and structural preservation.

Changes in discrimination-related performance show that larger representation dimensions are not always beneficial. When the dimension lies within a moderate range, latent representations achieve a reasonable balance between expressive power and discriminative constraints. This balance improves the system's ability to distinguish among different runtime states. When the dimension increases further, redundant features are gradually introduced. The latent space structure becomes more dispersed. This dispersion weakens the consistency of system behavior discrimination. The phenomenon suggests that excessive representational freedom can negatively affect behavior modeling.

From the overall trend of discrimination capability, the impact of the representation dimension on different performance aspects is not fully consistent. This difference reflects internal structural adjustments of system behavior under varying representation capacities. It indicates that unified representation influences more than individual discrimination outcomes. It also reshapes the global organization of system behavior in the latent space. With appropriate control of representation dimension, the system can maintain more stable behavior structures across different runtime phases. This contributes to improved reliability of overall monitoring.

With respect to false alarm-related behavior, variation in the representation dimension also has a significant effect on monitoring stability. An appropriate dimension allows normal operating states to form more

compact distributions in the latent space. This reduces the risk of misjudgment caused by local perturbations. When the dimension is excessively high, the looser latent structure tends to amplify irrelevant fluctuations. This leads to instability in monitoring signals. These results further demonstrate that selecting a suitable unified representation dimension is critical for building a reliable system behavior monitoring mechanism.

The figure below shows a comparison of the overall performance of different methods in terms of decision benefits, stability, and the quality of state and representation modeling, and the experimental results are shown in Figure 3.

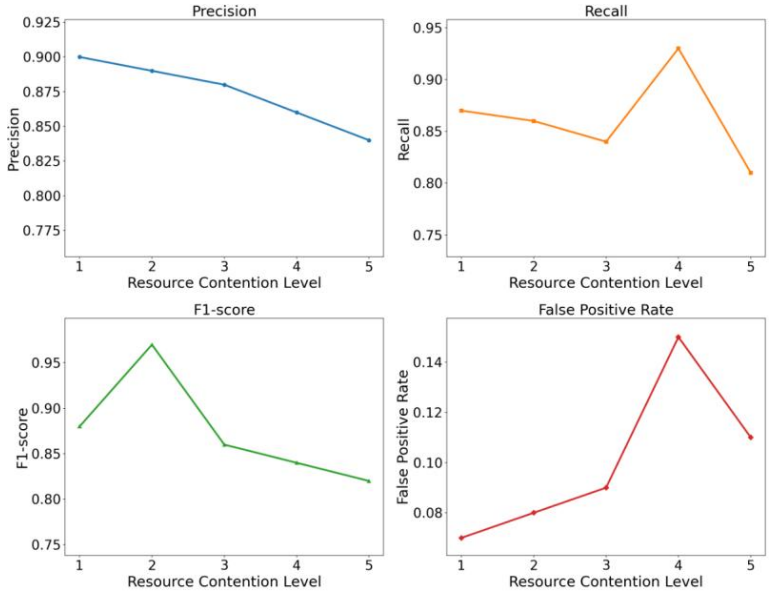

Figure 3. Analysis of the impact of changes in resource competition intensity on the stability and discriminative characteristics of system behavior monitoring.

The resource competition intensity rating of 1-5 represents a discrete five-level classification of the system's resource "strain" within the same time window, from low to high. Quantitatively, it uses the overall resource utilization range: Level 1 indicates low competition (comprehensive utilization of key resources such as CPU/GPU/memory is approximately 0-60%, with virtually no queuing); Level 2 indicates mild competition (approximately 60-75%, with some queuing but quickly resolved); Level 3 indicates moderate competition (approximately 75-85%, with frequent queuing and scheduling, and increased state fluctuations); Level 4 indicates high competition (approximately 85-95%, with resources nearing saturation and significant congestion and jitter); and Level 5 indicates extremely high competition (approximately 95-100%, with prolonged saturation accompanied by significant queuing/preemption/rate limiting). Therefore, the horizontal axis from 1 to 5 corresponds to a gradual transition from "resource surplus" to "near or near saturation" in the operating environment, used to evaluate the stability and quality of the monitoring method under different competitive pressures.

As resource contention intensifies, the intrinsic structure of system behavior gradually changes. This evolution places higher demands on the stability of unified behavior representations. In environments with low contention, system state evolution is relatively smooth. Behavior distributions remain well aggregated in the latent space. The monitoring mechanism, therefore, maintains consistent discriminative characteristics. When contention becomes stronger, interference among inference tasks increases significantly. System behavior exhibits higher volatility and uncertainty. This situation challenges behavior modeling and state discrimination. These observations indicate a direct impact of environmental complexity on the structural properties of system behavior.

From an overall monitoring perspective, changes in resource contention do not affect all discriminative properties in a uniform manner. Instead, contention influences monitoring outcomes indirectly by reshaping behavior distributions. Under certain contention levels, system behavior still forms relatively clear structural boundaries in the latent space. This supports effective discrimination by the monitoring mechanism. At higher contention levels, behavior patterns become more entangled. Monitoring signals are more sensitive to local disturbances. These results suggest that behavior learning based monitoring methods show a degree of adaptability in complex resource environments. Their stability, however, remains closely related to the intensity of runtime contention.

5. Conclusion

This work addresses the reliability of monitoring AI inference backends under complex operating conditions. From a system behavior modeling perspective, it proposes a learning driven monitoring approach. Unified representations and structured characterization of backend execution enable continuous perception and discrimination of system state variations. This study moves beyond the reliance of traditional monitoring methods on local metrics and static rules. It allows monitoring mechanisms to understand runtime characteristics from a holistic behavior perspective. It provides a more robust technical foundation for backend operation management in multi-tenant and multi-model concurrent scenarios. The proposed behavior modeling framework improves stability and controllability of inference services during long-term operation. It also introduces a new paradigm for automated operation and maintenance of intelligent infrastructure. Looking forward, this work establishes a basis for applying learning based behavior modeling to a broader range of backend management tasks. It has the potential to promote intelligent monitoring techniques in large-scale distributed inference services, cloud platforms, and emerging intelligent systems. It further offers sustained support for building more adaptive and trustworthy intelligent service backends.

References

[1] Crankshaw D, Wang X, Zhou G, et al. Clipper: A Low-Latency Online Prediction Serving System[C]//14th USENIX Symposium on Networked Systems Design and Implementation (NSDI). 2017: 613-627.

[2] Gujarati A, Karimi R, Alzayat S, et al. Serving DNNs like Clockwork: Performance Predictability from the Bottom Up[C]//14th USENIX Symposium on Operating Systems Design and Implementation (OSDI). 2020: 443-462.

[3] Romero F, Li Q, Yadwadkar N J, et al. INFaaS: Automated Model-less Inference Serving[C]//2021 USENIX Annual Technical Conference (USENIX ATC). 2021: 397-411.

[4] Pang G, Shen C, Cao L, et al. Deep Learning for Anomaly Detection: A Review[J]. ACM Computing Surveys, 2022, 54(2): 1-38.

[5] Siffer A, Fouque P A, Termier A, et al. Anomaly Detection in Streams with Extreme Value Theory[C]//Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2017: 1067-1075.

[6] Xu H, Chen W, Zhao N, et al. Unsupervised Anomaly Detection via Variational Auto-Encoder for Seasonal KPIs in Web Applications[C]//Proceedings of the Web Conference. 2018: 187-196.

[7] Zhang C, Yu M, Wang W, et al. MArk: Exploiting Cloud Services for Cost-Effective, SLO-Aware Machine Learning Inference Serving[C]//2019 USENIX Annual Technical Conference (USENIX ATC). 2019: 1049-1062.

[8] Rossi F, Nardelli M, Cardellini V. Horizontal and Vertical Scaling of Container-Based Applications Using Reinforcement Learning[C]//2019 IEEE 12th International Conference on Cloud Computing (CLOUD). IEEE, 2019: 329-338.

[9] Kang Y, Hauswald J, Gao C, et al. Neurosurgeon: Collaborative Intelligence Between the Cloud and Mobile Edge[C]//Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems. 2017: 615-629.

[10] Zhang X, Xu Y, Lin Q, et al. Robust Log-Based Anomaly Detection on Unstable Log Data[C]//Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 2019: 807-817.

[11]Mao H, Schwarzkopf M, Venkatakrishnan S B, et al. Learning Scheduling Algorithms for Data Processing Clusters[C]//Proceedings of the ACM Special Interest Group on Data Communication. 2019: 270-288.

[12]Ren H, Xu B, Wang Y, et al. Time-Series Anomaly Detection Service at Microsoft[C]//Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2019: 3009-3017.

[13]Tuli S, Casale G, Jennings N R. TranAD: Deep Transformer Networks for Anomaly Detection in Multivariate Time Series Data[J]. Proceedings of the VLDB Endowment, 2022, 15(6): 1201-1214.

[14]Du M, Li F, Zheng G, et al. DeepLog: Anomaly Detection and Diagnosis from System Logs Through Deep Learning[C]//Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. 2017: 1285-1298.

[15]Meng W, Liu Y, Zhu Y, et al. LogAnomaly: Unsupervised Detection of Sequential and Quantitative Anomalies in Unstructured Logs[C]//Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence. 2019: 4739-4745.

[16]Su Y, Zhao Y, Niu C, et al. Robust Anomaly Detection for Multivariate Time Series Through Stochastic Recurrent Neural Network[C]//Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2019: 2828-2837.

[17]Zhao H, Wang Y, Duan J, et al. Multivariate Time-Series Anomaly Detection via Graph Attention Network[C]//2020 IEEE International Conference on Data Mining (ICDM). IEEE, 2020: 841-850.

[18]Xu J, Wu H, Wang J, et al. Anomaly Transformer: Time Series Anomaly Detection with Association Discrepancy[C]//International Conference on Learning Representations. 2022.