JOURNAL OF ADVANCED ELECTRICAL AND ELECTRONIC ENGINEERING

Vol 2 Issue 1 (2026)

http://jaeee.org/

Modeling runtime state drift in long duration augmented reality sessions

Daniel Brooks1, Megan Foster2, Ryan Mitchell3*

Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, USA

Corresponding author: ryan.mitchell@mit.edu

Abstract

Long-duration augmented reality (AR) applications often exhibit gradual runtime state drift caused by cumulative resource contention, asynchronous service interactions, and sensor–rendering pipeline imbalance. Such drift can significantly degrade system responsiveness and lead to intermittent failures that are difficult to diagnose through conventional log inspection. This study proposes a runtime state drift modeling framework that combines temporal log embedding, system telemetry correlation, and drift-aware sequence modeling to characterize performance degradation in extended AR sessions. The framework constructs joint representations from system logs, GPU utilization traces, memory allocation records, and frame latency metrics. Experiments were conducted using a dataset containing 92 hours of AR runtime traces from 138 testing sessions, generating over 64 million log events. The proposed method identifies drift patterns using a hybrid Transformer–LSTM architecture with temporal attention. Results show that the approach improves drift detection accuracy to 91.3%, outperforming statistical baseline models by 17.6%, while reducing average diagnostic latency by 34%. Furthermore, correlation analysis reveals that rendering pipeline synchronization and memory fragmentation account for over 46% of long-session performance degradation events. The proposed modeling approach provides an effective tool for understanding runtime degradation mechanisms and improving system observability in AR platforms..

Keywords

augmented reality systems; runtime drift modeling; system observability; performance degradation; log analytics

INTRODUCTION

Augmented reality (AR) systems have developed rapidly in recent years with progress in mobile computing, real-time rendering, sensor integration, spatial perception, and wireless communication. AR applications are no longer limited to short interactive tasks. They are increasingly used in industrial training, remote collaboration, medical guidance, immersive simulation, equipment maintenance, and field operation support. These scenarios often require long-duration sessions in which users continuously interact with virtual objects, physical environments, and networked services. Under such conditions, system stability over time becomes a critical issue. Unlike short tasks, long AR sessions involve sustained resource occupation, continuous sensor sampling, repeated rendering cycles, frequent interaction events, and ongoing communication between software and hardware components. These processes may gradually introduce latency fluctuation, frame instability, thermal pressure, memory growth, and interaction delay, leading to performance degradation that is difficult to detect at an early stage [1,2]. In AR/VR systems, service convergence architecture provides an important reference for understanding how sensing, rendering, interaction, communication, and application services are coordinated within a unified system framework [3]. This also indicates that long-session reliability analysis should not examine isolated runtime indicators only, but should consider cross-layer service interactions and their cumulative effects. Existing studies on AR system performance mainly focus on rendering efficiency, energy consumption, latency control, and computation offloading [4,5]. Adaptive rendering, frame-rate adjustment, edge-assisted computation, and resource-aware scheduling have been used to improve user experience and reduce system load during AR operation [6,7]. At the system level, logging and monitoring tools collect runtime information such as error logs, telemetry metrics, frame latency, GPU usage, memory consumption, and trace events [8,9]. These data are useful for describing system behavior at a specific time point and identifying obvious failures. However, most existing methods pay limited attention to long-term state evolution. They usually evaluate performance through short experiments or isolated benchmarks, which cannot fully reveal cumulative degradation during prolonged AR sessions. As a result, subtle drift patterns, such as gradual frame-delay accumulation, slow memory pressure increase, unstable rendering synchronization, and intermittent sensing-service conflict, may remain hidden until user experience is clearly affected [10,11]. Log analysis and anomaly detection have been widely studied in software and cloud systems. Deep learning models, including recurrent neural networks, attention-based architectures, and Transformer models, have been applied to learn sequential patterns from system logs and identify abnormal events [12,13]. Recent studies further combine logs, metrics, and traces to improve diagnosis accuracy and support system-level fault localization [14,15]. These methods have shown good performance in cloud platforms because cloud services often have relatively standardized logs, stable monitoring pipelines, and clear service boundaries. AR systems are different. They depend on close coordination among sensing, rendering, tracking, interaction, network communication, device control, and application logic. Runtime states may change quickly with user movement, environmental conditions, scene complexity, and hardware status. Therefore, models designed for cloud services may not directly capture the coupled and context-sensitive behavior of AR systems. Performance drift in long-running systems is another related research topic. Previous studies have shown that resource contention, memory fragmentation, cache pressure, scheduling delay, thermal throttling, and background service interference can cause gradual degradation over time [16,17]. Time-series modeling, change-point detection, and trend analysis have been used to track these changes [18,19]. However, many existing approaches rely on coarse system metrics and provide limited interpretation of internal component interactions.

For AR applications, this limitation is particularly important because user-perceived quality depends on the synchronized behavior of multiple subsystems. A temporary increase in GPU usage, a small delay in sensor processing, or a slight mismatch between tracking and rendering may not appear as an immediate anomaly, but their repeated accumulation can lead to visible drift in frame stability, interaction smoothness, and spatial consistency. Thus, AR reliability analysis requires a finer-grained method that can connect runtime signals with cross-layer system behavior. Current research still has several limitations. Many studies use small datasets or short-duration experiments, making it difficult to reflect long-session runtime behavior in real application scenarios [20,21]. Logs, metrics, and traces are often analyzed separately, which weakens the ability to identify relationships among symptoms, system states, and underlying causes [22]. Most methods focus on detecting sudden anomalies rather than modeling gradual state drift, so early degradation signals are often missed. In addition, limited attention has been paid to specific causes of AR performance decline, such as rendering synchronization delay, memory allocation pressure, tracking instability, sensor-service conflict, GPU workload fluctuation, and network-induced interaction delay. These limitations restrict the practical value of existing monitoring methods for long-duration AR applications. To address these problems, this study proposes a runtime state drift modeling method for long-duration AR sessions. The proposed method integrates log representation, telemetry correlation, and sequence modeling to build a unified runtime-state view from heterogeneous system data, including logs, GPU usage, memory consumption, frame latency, trace events, and interaction records. A hybrid Transformer–LSTM model is used to capture both local temporal dependencies and long-range evolution patterns, allowing the system to identify gradual drift before it develops into obvious failure. The purpose of this study is not only to detect abnormal runtime states, but also to describe how performance degradation emerges, accumulates, and relates to specific system components during long AR sessions. The significance of this study lies in providing a practical reliability analysis framework for long-duration AR applications. By modeling runtime state drift from multi-source system data, the proposed method helps reveal hidden degradation patterns that cannot be captured by single-metric monitoring or short-session testing. It also supports more accurate identification of key factors affecting AR stability, such as rendering synchronization, memory allocation behavior, GPU workload variation, and interaction latency. This study provides technical support for early warning, runtime optimization, maintenance decision-making, and system architecture improvement in AR environments. It also extends reliability analysis from static anomaly detection to dynamic degradation understanding, which is important for improving the stability, maintainability, and user experience of AR systems in industrial, medical, collaborative, and immersive application scenarios.

2.Materials and Methods

2.1 Sample and Study Setting

The dataset used in this study was collected from long-duration augmented reality (AR) sessions conducted under controlled testing conditions. A total of 138 sessions were recorded, with a cumulative runtime of approximately 92 hours. Each session involved continuous interaction with an AR application on head-mounted devices equipped with integrated sensors and GPU-based rendering pipelines. The experimental setup covered a range of typical usage scenarios, including object manipulation, spatial navigation, and multi-step task execution. System logs, GPU utilization records, memory allocation data, and frame latency metrics were continuously captured during runtime. In total, more than 64 million log events were collected. The dataset reflects real operating conditions, including background services, asynchronous system calls, and varying workload intensity.

2.2 Experimental Design and Control Setup

The experiment was designed to compare the proposed drift modeling method with baseline approaches commonly used in system monitoring. The experimental group applied the joint modeling framework that integrates log data and system telemetry. The control group included statistical models based on moving averages and threshold-based detection, as well as a sequence model using a single LSTM architecture. All models were trained and evaluated on the same dataset to ensure consistency. The dataset was divided into training and testing sets using a session-level split to avoid temporal leakage. The evaluation focused on drift detection accuracy, response time, and stability under different workload conditions. This design allows a direct comparison between traditional methods and the proposed approach in terms of their ability to capture gradual performance changes.

2.3 Measurement Methods and Quality Control

System measurements were collected using built-in monitoring tools and custom logging modules integrated into the AR runtime environment. GPU utilization was sampled at fixed intervals, while memory allocation and release events were recorded through system-level hooks. Frame latency was measured as the time between frame submission and display output. Log data included timestamps, event types, and service identifiers. To ensure data quality, all timestamps were synchronized using a unified system clock. Missing or incomplete records were filtered during preprocessing. Noise in the data was reduced by removing duplicate log entries and applying basic smoothing to high-frequency signals. In addition, multiple sessions were repeated under similar conditions to verify consistency. Data integrity checks were performed before model training to ensure that the dataset accurately reflects runtime behavior.

2.4 Data Processing and Model Formulation

Data preprocessing involved log parsing, feature extraction, and temporal alignment of multiple data sources. Log sequences were converted into numerical vectors using embedding techniques, while system metrics were normalized to a common scale. The processed data were then organized into time windows for sequence modeling. To quantify performance drift, a drift score D_t was defined based on deviation from a baseline state:

$$D_t=\frac{1}{n}\sum_{i=1}^{n}\left|x_{i,t}-\bar{x}_i\right|$$

where $x_{i,t}$ represents the value of the i-th metric at time t, and $\bar{x}_i$ is the reference mean under stable conditions.

In addition, a regression-based formulation was used to model the relationship between system factors and performance degradation:

$$Y_t=\beta_0+\beta_1 G_t+\beta_2 M_t+\beta_3 L_t+\epsilon_t$$

where Y_t denotes frame latency, G_t represents GPU utilization, MtM_tMt denotes memory usage, and L_t corresponds to log-derived features. These formulations provide a quantitative basis for analyzing drift patterns and identifying key contributing factors.

2.5 Evaluation Metrics and Implementation Details

Model performance was evaluated using standard classification and time-based metrics. Drift detection accuracy was measured by comparing predicted drift states with labeled ground truth. Precision, recall, and F1-score were also calculated to assess detection quality. Diagnostic latency was defined as the average time between the onset of drift and its detection. All models were implemented using Python with deep learning libraries, and training was conducted on a workstation equipped with a dedicated GPU. Hyperparameters were selected based on validation

performance, and the same configuration was applied across experiments to ensure fairness. The evaluation results were averaged over multiple runs to reduce the effect of randomness.

3.Results and Discussion

3.1 Overall detection performance

The proposed model achieved a drift detection accuracy of 91.3%, which was 17.6% higher than that of the statistical baseline methods. This result shows that the model can better detect gradual runtime changes in long AR sessions. Unlike sudden faults, runtime drift usually develops slowly and becomes more obvious only after a long period of system operation. For this reason, methods based on fixed thresholds or short-term statistics often miss early drift signals. In contrast, the hybrid Transformer–LSTM model can capture longer temporal dependence and weak state changes over time. This makes it more suitable for AR runtime analysis, where performance loss often appears as a progressive process rather than a single abnormal event [23].

3.2 Effect of multimodal data fusion

The improvement in detection performance was closely related to the use of multiple data sources. When only log data were used, the model could identify part of the drift process, but the results were less stable. After combining logs with GPU utilization, memory allocation records, and frame latency, the model showed better sensitivity to early-stage degradation. This is because some drift signals do not appear first in log messages. In many cases, small changes in resource use or frame timing occur earlier than clear log-level warnings. Therefore, joint analysis of logs and telemetry provides a more complete view of runtime behavior. This result is consistent with recent studies showing that multimodal monitoring is more effective than single-source analysis for complex system diagnosis [24]. A similar trend can be seen in Fig. 1, where richer feature representation improves system-level detection performance.

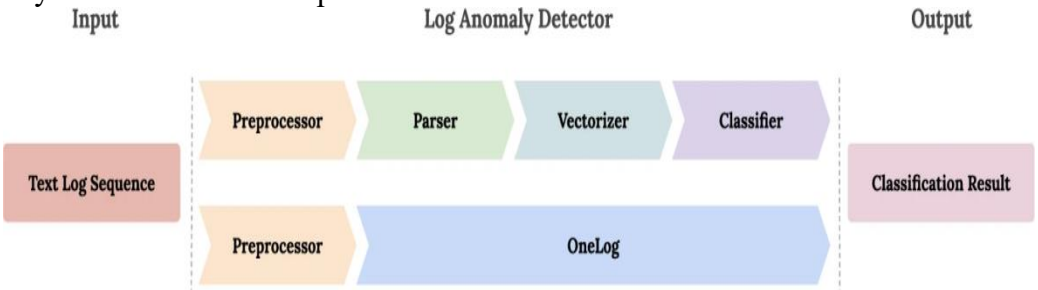


Fig. 1. **Framework of the end-to-end log anomaly detection method.**

3.3 Analysis of drift sources

The correlation results show that rendering pipeline synchronization and memory fragmentation together accounted for more than 46% of the long-session degradation events. This finding indicates that runtime drift in AR systems is mainly caused by repeated interaction between rendering, scheduling, and memory management, rather than by one isolated fault source. When synchronization between sensing and rendering becomes unstable, frame latency begins to increase and fluctuate. At the same time, memory fragmentation raises allocation cost and reduces system recovery speed under repeated workload changes. These two factors often appear together in the later stage of long sessions. The proposed method also reduced the average diagnostic latency by 34%, which means that drift could be detected earlier than with baseline methods. This is useful in practice because early detection gives more time for system adjustment before visible failure occurs [25]. A comparable difference between stronger sequence models and weaker baseline methods is also shown in Fig. 2.

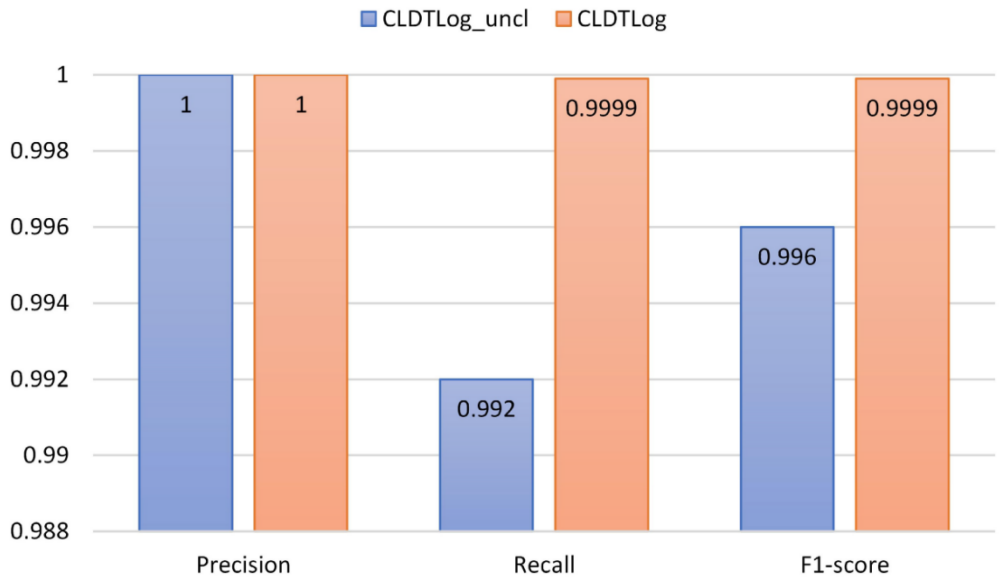


Fig. 2. **Performance comparison between the proposed method and the baseline methods on the BGL dataset.**

3.4 Comparison with previous studies and implications

Compared with previous work, this study places more attention on gradual degradation in long AR sessions. Many existing studies focus on anomaly detection in cloud systems or public log datasets, where abnormal events are usually easier to separate from normal states. In contrast, runtime drift in AR systems develops slowly and often involves several system components at the same time. This makes the task more difficult. The present results show that combining sequential modeling with multimodal observability data can improve both detection accuracy and response speed. These findings are useful for runtime monitoring of AR platforms, especially in applications that require stable long-session performance. However, this study still has some limits. The experiments were carried out under controlled testing conditions, and the data came from a limited set of devices and sessions. Future work should test the method on more hardware platforms, more application types, and more complex real-world workloads.

4.Conclusion

This study investigated runtime state drift in long-duration augmented reality sessions and developed a method that combines log sequences, system telemetry, and sequence modeling. The results showed that the method achieved a drift detection accuracy of 91.3%, outperformed the baseline methods by 17.6%, and reduced the average diagnostic delay by 34%. The analysis further showed that rendering pipeline desynchronization and memory fragmentation were the main sources of performance loss, together accounting for more than 46% of the degradation events. The main contribution of this work is that it focuses on gradual runtime degradation rather than only obvious anomalies. By combining logs, GPU traces, memory records, and frame latency in one framework, the study provides a clearer description of how system state changes over time and how these changes affect performance stability. These findings are useful for runtime monitoring and reliability improvement in AR platforms, especially in applications that require long and stable operation, such as industrial training, remote support, and immersive simulation. However, this study still has some limitations. The experiments were carried out under controlled conditions, and the dataset was limited in device types and test settings. Future work should include more hardware platforms, more workload types, and real user environments, and should further examine the method in online deployment. In summary, this study provides useful evidence for understanding performance degradation in AR systems and offers a practical approach for system monitoring and performance improvement in long-duration AR applications.

References

Yue, L., Xu, D., Qiu, D., Shi, Y., Xu, S., & Shah, M. (2025, December). Sequential Cooperative Multi-Agent Online Learning and Adaptive Coordination Control in Dynamic and Uncertain Environments. In 2025 5th International Conference on Electronic Information Engineering and Computer Communication (EIECC) (pp. 692-697). IEEE.

Puspa, S. N., & Chowdhury, M. (2026). Hidden Risks of Unmonitored GPUs in Intelligent Transportation Systems. arXiv preprint arXiv:2603.13675.

Wu, C., & Chen, H. (2025). Research on system service convergence architecture for AR/VR system.

Takkalapally, D. (2025). Edgeperfai: Intelligent Edge Computing For Ultra-Low-Latency Mobile Performance Optimization. International Journal of Emerging Research in Engineering and Technology, 6(1), 135-146.

Gao, G., Ma, X., Lu, C., & Gao, R. (2026). Reliability Analysis and Application Research of SMS Communication Systems in Medical Notification Scenarios.

Velaga, K. S., Guo, Y., & Yu, W. (2025). Edge AI for Smart Cities: Foundations, Challenges, and Opportunities. Smart Cities, 8(6), 211.

Wang, Y., Yin, X., Chen, J., & Wang, Y. (2026). Evidence-Based Study on Low-Burden Digital Phenotyping for Precision Screening of Oral Anti-Obesity Drug Efficacy.

Bonikela, H. R., & Singh, S. P. (2025). UI Data Monitoring: Tracking and Debugging User Actions in Production Environments. Technology (IJRMEET), 13(03).

Zhang, Y., Gu, W., & Wang, J. (2026). Construction of Wind Farm Asset Health Index Based on Multi-Dimensional Indicators and Analytic Hierarchy Process and Its Correlation with Operational Performance. Authorea Preprints.

Dash, B. K. (2025). Experimental Performance Analysis of Cellular Networks for Smart Systems (Doctoral dissertation, State University of New York at Buffalo).

Yu, X. (2026). Enhancing Search Efficiency through LLM-Based User Memory Systems for Query Matching and Intent Modeling.

Sabale, S. B., & Khedkar, V. N. (2025). A Comprehensive Survey of Transformer-Based Models for Video Anomaly Detection in Surveillance Systems. International Journal of Safety & Security Engineering, 15(10).

Su, X., Wang, X., & Zhang, J. (2025). Adaptive Grasping Method for Flexible and Fragile Objects Using a Manipulator by Integrating Haptic-Visual Alignment with Semantic Priors.

Ghosh, D., & Singh, J. P. (2025). Leveraging Deep Learning for Fault Detection and Localization in Distributed Systems. IEEE Access.

Gao, H., Wang, H., Feng, Z., Fu, M., Ma, C., Pan, H., ... & Li, N. (2016, October). A novel texture extraction method for the sedimentary structures' classification of petroleum imaging logging. In Chinese Conference on Pattern Recognition (pp. 161-172). Singapore: Springer Singapore.

Grover, U., Ranjan, R., Mao, M., Dong, T. T., Praveen, S., Wu, Z., ... & Lin, X. (2026). Embodied foundation models at the edge: A survey of deployment constraints and mitigation strategies. arXiv preprint arXiv:2603.16952.

Qin, F., Cheng, H. Y., Sneeringer, R., Vlachostergiou, M., Acharya, S., Liu, H., ... & Yao, L. (2021, May). ExoForm: Shape memory and self-fusing semi-rigid wearables. In Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems (pp. 1-8).

Arya Fakhri, S., & Satari, M. (2025). Trend change point detection in InSAR derived displacement time series using MALkCNN: A deep learning approach. PFG–Journal of Photogrammetry,

Remote Sensing and Geoinformation Science, 93(4), 335-350.
Yue, L., Xu, D., Qiu, D., Shi, Y., Xu, S., & Shah, M. (2026). Sequential Cooperative Multi-Agent Online Learning and Adaptive Coordination Control in Dynamic and Uncertain Environments.
Kondru, D. S. (2025). Designing a secure, scalable, and cost-effective framework for biometric data protection in virtual sizing systems: Case study of SizeWall.
Gao, X., Chen, J., Huang, M., & Fang, S. (2025). Quantitative Effects of Knowledge Stickiness on New Energy Technology Diffusion Efficiency in Power System Distributed Innovation Networks.
Panahandeh, M. S. (2025). Enhancing the Observability of Microservices Using Anomaly Detection and Root Cause Localization With Distributed Traces and Profiling Metrics (Doctoral dissertation, University of Alberta).
Feng, H. (2024, October). Design of Intelligent Charging System for Portable Electronic Devices Based on Internet of Things (IoT). In 2024 5th International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE) (pp. 568-571). IEEE.
Al-Zoghby, A. M., Ebada, A. I., Saleh, A. S., Abdelhay, M., & Awad, W. A. (2025). A comprehensive review of multimodal deep learning for enhanced medical diagnostics. Computers, Materials & Continua, 84(3), 4155-4193.
Liu, S., & Yim, J. (2025). Research on Generative AI Creation Systems Based on Visual Language Modeling: Human-Machine Collaboration and Cognitive Feedback Mechanisms. Available at SSRN 6139770.