IRSTI 28.23.37 https://doi.org/10.26577/jpcsit4220265

Yesset Zhussupov[1], Ainur Zhumadillayeva[1*], Shona Shinassylov[2], Ainur Amirtayeva[2]

[1]L.N. Gumilyov Eurasian national university, Astana, Kazakhstan
[2]Al-Farabi Kazakh National University, Almaty, Kazakhstan
*e-mail: zhumadillayeva_ak@enu.kz

COMPARATIVE ANALYSIS OF SEQUENCE-BASED AND GRAPH-BASED DEEP LEARNING FOR ANOMALY DETECTION IN MICROSERVICE TRACES

Abstract. The transition from monolithic to microservice architectures has significantly increased the complexity of system observability. Distributed tracing has emerged as an indispensable instrument for monitoring inter-service interactions, yet traditional anomaly detection methods often fail to capture the complex structural relationships inherent in these systems. This paper presents a comparative study of sequence-based approaches, specifically Long Short-Term Memory (LSTM) networks, and graph-based deep learning methods such as Graph Neural Networks (GNNs) for anomaly detection in microservice traces. The comparison combines a structured analysis of existing literature with original experimental evaluation, in which three representative models – DeepLog (LSTM), DeepTraLog (GGNN), and TraceVAE (Graph VAE) – are trained and evaluated on the TrainTicket benchmark comprising 106,312 traces from 35 microservices. We evaluate these paradigms across three dimensions: structural awareness, detection accuracy, and computational efficiency. Our experimental results, together with findings reported in prior work, confirm that graph-based models achieve higher detection accuracy (F1 up to 0.896) compared to sequence-based alternatives (F1 = 0.758), while also revealing practical trade-offs in training cost, memory consumption, and deployment complexity.

Keywords: microservice architecture, anomaly detection, graph neural networks, LSTM, distributed tracing, deep learning, observability.

1. Introduction

Modern industrial microservice systems often comprise thousands of services running across distributed containers and virtual machines [1], [2]. In contrast to conventional monolithic designs, where application logic is encapsulated in a single binary, microservices rely on loosely coupled, autonomous units communicating via lightweight protocols such as Remote Procedure Calls (RPC) or Representational State Transfer (REST) APIs [3], [4]. This architectural paradigm shift provides substantial benefits in terms of scalability, deployment flexibility, and team autonomy. However, it simultaneously introduces substantial "observability challenges," where a failure in one service can propagate through complex invocation chains, causing cascading effects across the entire distributed system [5], [6].

To mitigate these operational risks, distributed tracing technologies such as OpenTracing, Jaeger, and Apache SkyWalking have been developed to record the end-to-end execution paths of user requests as a series of hierarchically organized "spans" [7], [8]. Each span represents a unit of work within the system, containing metadata about timing, service identity, and parent-child relationships. However, the telemetry data generated by these systems is often high-dimensional, semi-structured, and characterized by complex non-linear dependencies, rendering manual troubleshooting and simple rule-based detection approaches insufficient for modern production environments [9], [10].

The emergence of machine learning (ML) and deep learning (DL) techniques has offered promising solutions for automated anomaly detection in distributed systems. Among these approaches, two paradigms have attracted considerable scholarly interest: sequence-based models, particularly Long Short-Term Memory (LSTM) networks, and graph-based models, including various Graph Neural Network (GNN) architectures. This paper offers an exhaustive evaluation of these two paradigms, examining their relative strengths and limitations across multiple evaluation dimensions.

The contribution of this paper is twofold. First, we provide a structured analytical comparison of

sequence-based and graph-based paradigms, synthesizing detection accuracy and computational efficiency findings reported across multiple independent studies. Second, we complement this literature-based analysis with original experimental evaluation: three representative models – DeepLog, DeepTraLog, and TraceVAE – are trained and tested on the same TrainTicket dataset comprising 106,312 traces from 35 microservices. Results that originate from our experiments are explicitly distinguished from those reported in prior work. This mixed-methods strategy addresses a methodological gap identified in the existing literature, where cross-study comparisons of anomaly detection models are complicated by differences in datasets, preprocessing, hardware, and evaluation protocols.

The remainder of this paper is organized as follows. Section II provides background on microservice architectures and distributed tracing. Section III examines the structural awareness capabilities of both approaches. Section IV presents detection accuracy benchmarks, including our unified experimental setup and results. Section V analyzes computational efficiency considerations. Section VI discusses limitations, implications, and future directions, and Section VII concludes the paper.

2. Background and Related work

A. Microservice Architecture and Observability

Microservice architecture represents a fundamental departure from traditional monolithic application design. In a monolithic system, all functional components share a single codebase and are deployed as a unified artifact. While this approach simplifies initial development and deployment, it creates significant challenges for scaling, maintenance, and technology evolution [11]. Microservice architecture addresses these limitations by decomposing applications into small, independently deployable services, each responsible for a specific business capability [12].

However, this decomposition comes at the cost of increased operational complexity. A single user request may traverse dozens or even hundreds of services before completion, with each service potentially introducing its own failure modes, latency characteristics, and resource constraints [13]. The challenge of understanding system behavior in this context-commonly referred to as "observability" – has become a critical concern for organizations operating at scale.

B. Distributed Tracing Fundamentals

Distributed tracing provides a mechanism for tracking request flow across service boundaries. The core abstraction is the "trace," which represents the complete journey of a request through the system. Each trace consists of multiple "spans," where each span represents a named, timed operation within a single service [7]. Spans are organized hierarchically, with parent-child relationships indicating invocation patterns.

Modern tracing systems propagate context information (typically including a trace ID and span ID) through request headers, enabling the reconstruction of complete request paths even in highly distributed environments. This contextual information, combined with timing data and service metadata, provides the foundation for various analytical approaches including anomaly detection, performance optimization, and root cause analysis [14], [15].

C. Traditional Anomaly Detection Approaches

Early approaches to anomaly detection in distributed systems relied primarily on statistical methods and rule-based systems. These approaches typically involved setting thresholds for metrics such as response time, error rates, or resource utilization. While effective for detecting gross violations, such approaches struggle with the dynamic and context-dependent nature of modern microservice systems. The advent of machine learning has enabled more sophisticated detection methods capable of learning complex patterns from historical data.

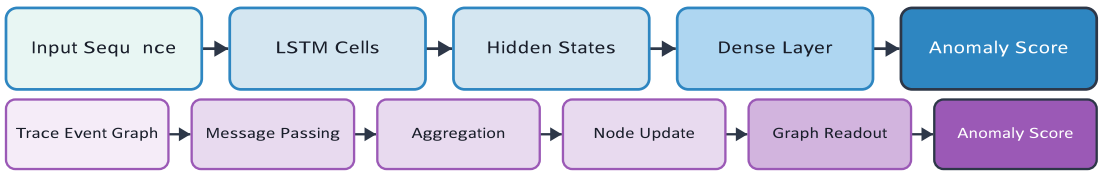
3. Structural awareness: Sequences vs. Hierarchies

The fundamental difference between sequence-based and graph-based models lies in their perception and representation of trace structure. This distinction has profound implications for anomaly detection capability, particularly in systems with complex invocation patterns.

A. Sequence-based Modeling with LSTM Networks

Traditional sequence models such as DeepLog and Multimodal LSTM treat traces as linear sequences of service invocations or log events [16], [17]. These models operate under the assumption of a sequential temporal flow where event A leads to event B in a strictly ordered manner. Long Short-Term Memory networks are particularly well-suited for this paradigm due to their ability to capture long-range dependencies through their gating mechanisms [18].

Figure 1
Architectural comparison: (a) Sequence-based LSTM pipeline, (b) Graph-based GNN pipeline



However, microservice traces are inherently tree-structured rather than linear [19]. This structural mismatch creates several fundamental limitations for sequence-based approaches:

- Invocation Hierarchy: A parent span may initiate multiple child spans simultaneously, representing parallel execution paths. Sequence models cannot naturally represent this branching structure [20].

- Asynchronous and Parallel Calls: Microservices frequently utilize parallel execution to reduce latency. In a sequence-based model, log events from parallel spans interleave in arbitrary orders, leading the model to falsely report unseen subsequences as anomalies [21].

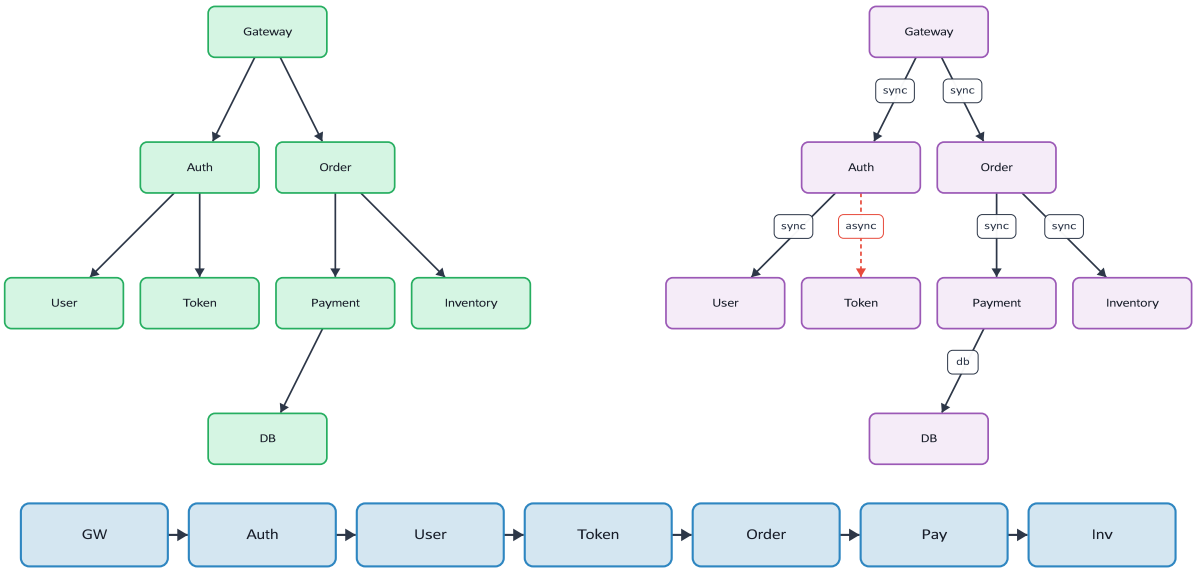
- Context Loss: When flattening a tree structure into a sequence, critical contextual information about the relationship between operations is lost, potentially masking important anomaly indicators.

B. Graph-based Modeling with Graph Neural Networks

Graph-based frameworks address these limitations by preserving the natural structure of distributed traces. Systems such as DeepTraLog utilize a Trace Event Graph (TEG) representation that unifies inter-service interactions (traces) and intra-service behaviors (logs) into a single cohesive structure [22]. In this paradigm, the trace is represented as a directed graph where nodes correspond to spans or log events and edges represent relationships between them.

Graph representations enable the explicit encoding of multiple relationship types, including: (1) Sequence relationships between consecutive events within a service; (2) Synchronous request relationships between caller and callee; (3) Synchronous response relationships for return paths; and (4) Asynchronous request relationships for fire-and-forget patterns [23]. This rich relational encoding provides models with the structural context necessary for accurate anomaly detection.

Figure 2
Trace representations: (a) Original tree structure (b) Graph with typed edges (structure preserved)., (c) Flattened sequence (information lost)



By modeling traces as graphs, GNNs-specifically Gated Graph Neural Networks (GGNNs)-can capture the "back-pressure" and cascading effects that propagate along service dependencies [24]. This capability is essential for detecting anomalies that manifest as structural deviations rather than simple metric threshold violations.

Advanced graph-based models like TraceVAE further extend this approach by utilizing a dualvariable architecture that encodes structural and temporal features separately [25]. This separation ensures that the model understands both the topological context of each service invocation and its temporal characteristics, enabling more nuanced anomaly detection.

4. Detection accuracy: Empirical evidence

Empirical evidence from multiple research studies indicates a significant performance gap between sequence-based and graph-based approaches for microservice anomaly detection. This section presents quantitative comparisons and discusses the underlying factors contributing to these differences.

A. Benchmark Methodology

The TrainTicket microservice system has emerged as a widely-used benchmark for evaluating anomaly detection approaches [26]. This system simulates a realistic ticket booking application comprising dozens of interconnected microservices. Researchers inject various fault types including service failures, latency anomalies, and resource exhaustion to evaluate detection capabilities across different anomaly categories.

B. Unified Experimental Setup

To address the methodological limitation of cross-study comparison, we conduct original experiments in which three representative models are evaluated under controlled conditions. We select DeepLog [16] as the sequence-based baseline, DeepTraLog [22] as the primary graph-based model, and TraceVAE [25] as an alternative graph-based architecture employing variational inference.

All models are trained and evaluated on traces extracted from the TrainTicket microservice benchmark [26], comprising 106,312 traces from 35 services (89,724 normal, 16,588 anomalous). The traces originate from 14 fault injection scenarios (F01–F14) covering service failures, latency anomalies, and resource exhaustion. We apply a consistent split: 62,806 training traces, 10,630 validation traces (8,972 normal + 1,658 anomalous), and 32,876 test traces (17,946 normal + 14,930 anomalous).

A unified data conversion pipeline transforms the raw traces into the input format required by each model. DeepLog and DeepTraLog experiments were conducted on an NVIDIA A100SXM4-40GB GPU; TraceVAE was evaluated on a Tesla T4 GPU due to dependency constraints of the DGL framework version required by its implementation. While this hardware difference limits direct comparison of computational efficiency between TraceVAE and the other models, detection accuracy metrics (Precision, Recall, F1) remain comparable as they are hardware-independent. We report all metrics using identical evaluation procedures across models.

C. Quantitative Performance Comparison

Table 1 presents comparative results from recent evaluation studies. As shown, graph-based models consistently outperform their sequence-based counterparts across precision, recall, and F1score metrics.

Table 1

Performance Comparison of Anomaly Detection Models. Results marked "This work" were obtained on 106,312 TrainTicket traces (62,806 train / 10,630 val / 32,876 test). DeepLog and Deep-TraLog used NVIDIA A100-SXM4-40GB; TraceVAE used Tesla T4 due to framework constraints.

Literature results are reported as published

Model	Type	Dataset	Hardware	P	R	F1	Source
TraceVAE	GNN+VAE	TrainTicket	Tesla T4	0.940	0.855	0.896	This work
DeepTraLog	GNN (GGNN)	TrainTicket	A100-40GB	0.937	0.807	0.868	This work
DeepLog	LSTM	TrainTicket	A100-40GB	0.688	0.844	0.758	This work
DeepTraLog	GNN (GGNN)	TrainTicket	2×V100	0.930	0.978	0.954	[22]
TraceVAE	GNN+VAE	TrainTicket	N/R	0.912	0.956	0.933	[25]

Continuation of the table

Model	Type	Dataset	Hardware	P	R	F1	Source
TCN-VAE	Hybrid	TrainTicket	N/R	0.938	0.927	0.933	[34]
Eadro	Multi-modal	TrainTicket	N/R	0.891	0.912	0.901	[34]
GRU-SVDD	Sequence	TrainTicket	N/R	0.834	0.803	0.818	[22]
Sleuth	Graph	N/R	N/R	0.856	0.789	0.821	[36]
DeepLog	Sequence	HDFS/BGL	N/R	0.762	0.721	0.741	[16]
LogAnomaly	Sequence	HDFS/BGL	N/R	0.724	0.698	0.711	[19]
MLP Baseline	Vector	TrainTicket	N/R	0.681	0.645	0.663	[25]
TraceAnomaly	Vector	TrainTicket	N/R	0.412	0.267	0.321	[9]

Figure 3
Performance comparison: (a) Detection accuracy metrics, (b) F1-score ranking by model type

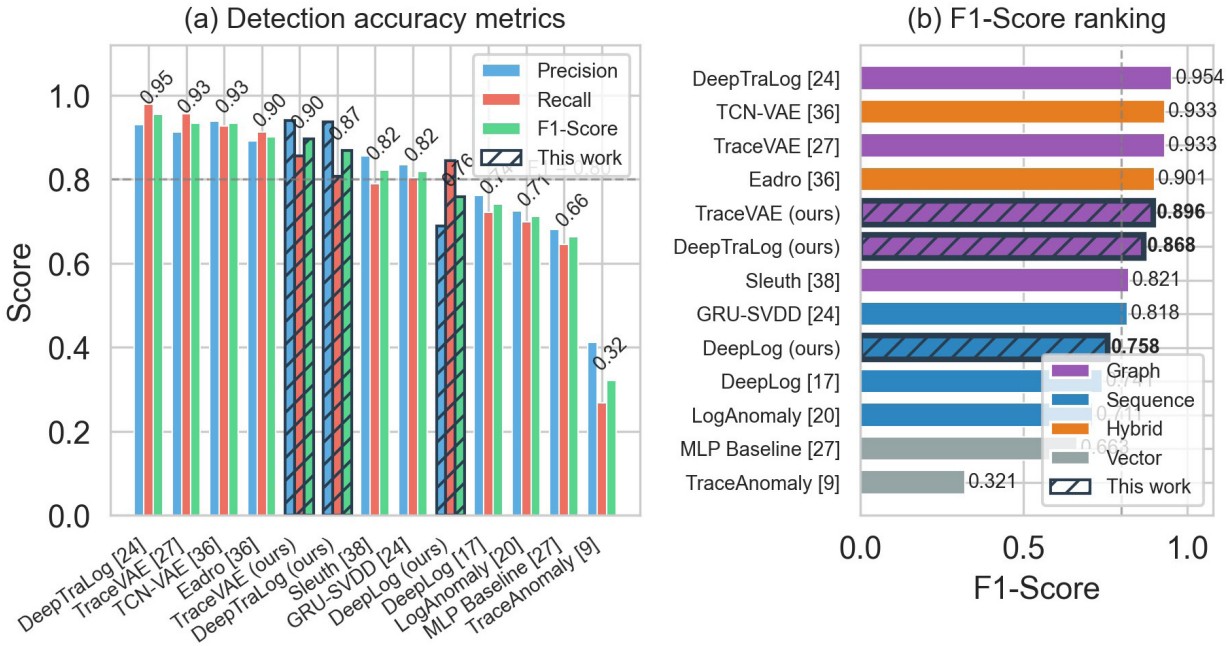
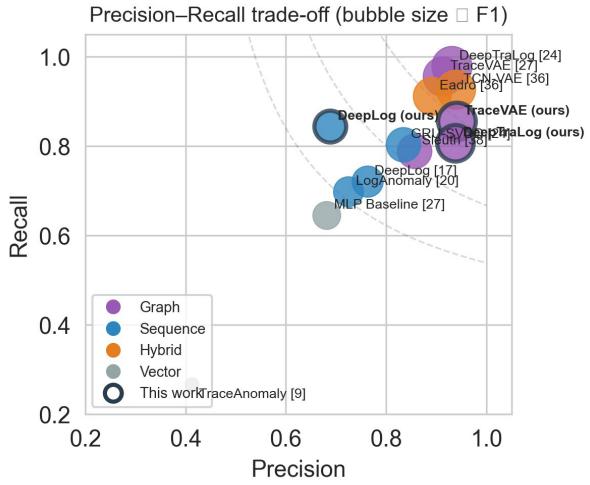


Table 1 presents both our experimental results and findings from the literature. In our unified evaluation, TraceVAE achieved the highest F1-score (0.896), followed by DeepTraLog (0.868), both substantially outperforming the LSTM-based DeepLog (0.758). Notably, our results differ from those reported in the original studies – for example, DeepTraLog achieves F1=0.954 in [22] versus 0.868 in our experiments – illustrating the sensitivity of these models to data splits, preprocessing pipelines, and hardware configurations. This discrepancy reinforces the need for unified benchmarking as presented in this work.

Graph-based models like TraceVAE address advanced probabilistic issues that particularly affect sequence-based approaches. One such issue is Negative Log-Likelihood (NLL) Inversion, where sequential models suffer from an "entropy gap" [25]. In this phenomenon, lower-dimensional anomalies-such as a missing node in a large trace-paradoxically result in lower anomaly scores than normal traces, leading to false negatives.

Figure 4
Precision–Recall trade-off across all evaluated models. Bubble size is proportional to F1score; bold edges indicate results from this work



D. Addressing Probabilistic Challenges

GNN-based approaches mitigate this issue through techniques such as Node Count Normalization and Gaussian Std-Limit within the graph framework [25], [27]. These methods effectively normalize for trace complexity, ensuring that anomalies are detected regardless of trace size. This capability is particularly important in production environments where trace complexity varies significantly across different request types.

5. Computational Efficiency Analysis

Microservice telemetry is increasingly high-dimensional, with individual traces potentially involving hundreds of service calls and thousands of log events [1]. Computational efficiency is therefore a critical consideration for production deployment of anomaly detection systems.

A. Training and Inference Latency

Empirical analysis reveals an interesting trade-off between training time and testing efficiency. In our unified experiments on the TrainTicket benchmark (NVIDIA A100 GPU), DeepTraLog processed the test set of 32,876 traces in 13.2 seconds compared to 3,115.1 seconds for DeepLog, yielding a 237× difference in inference time. This substantial speedup, obtained under controlled conditions with identical data and hardware, exceeds the approximately 40× improvement reported in prior cross-study comparisons [22] and suggests that the efficiency advantage of graph-based inference may be more pronounced than previously documented. GGNNs treat events as parallel network units amenable to efficient message-passing algorithms, whereas LSTMs must serially process every event in a sequence [28], [29].

Table 2
Training and Inference Time Comparison. Inference times are for the full test set (32,876 traces for "This work" rows). Times are not directly comparable across different hardware

Model	Type	Hardware	Training Time	Inference Time	Source
DeepTraLog	GNN (GGNN)	A100-40GB	1,618 s (~27 min)	13.2 s	This work
TraceVAE	GNN+VAE	Tesla T4	27,112 s (~7.5 h)	281.9 s	This work
DeepLog	LSTM	A100-40GB	7,584 s (~2.1 h)	3,115.1 s	This work
DeepTraLog	GNN (GGNN)	2×V100	~67 min	77 s	[22]
TraceVAE	GNN+VAE	N/R	~85 min	92 s	[25]
LogAnomaly	Sequence	N/R	~45 min	3,136 s	[19]
DeepLog	Sequence	N/R	~25 min	~800 s	[16]

Figure 5
Computational efficiency comparison: (a) Training time, (b) Inference time showing 237× speedup for DeepTraLog vs DeepLog

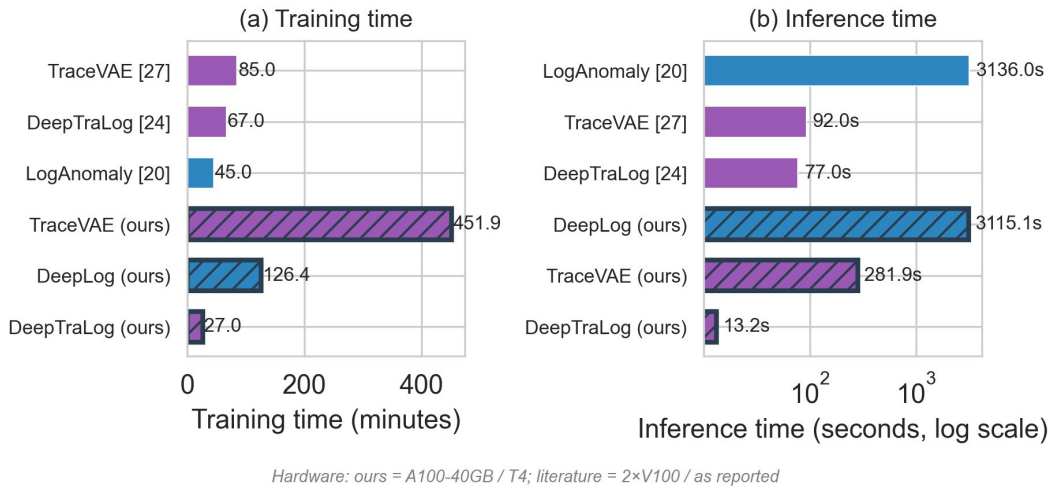


Figure 6
AUC-ROC comparison for models evaluated in this work

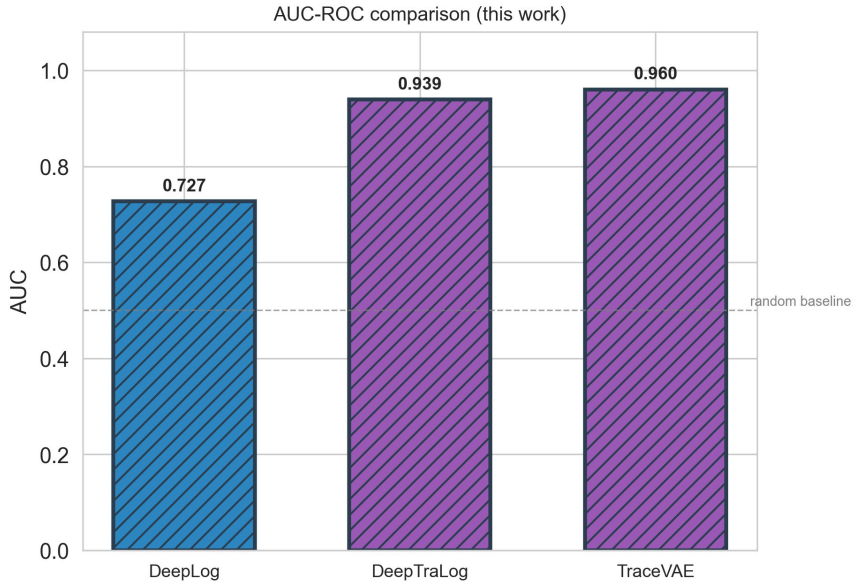


Table 3
Model Characteristics

Capability	DeepTraLog	TraceVAE	DeepLog	LogAnomaly
Structural Awareness	Full	Full	None	None
Parallel Call Handling	Yes	Yes	No	No
Log+Trace Integration	Combined	Trace only	Log only	Log only
Unsupervised	No	Yes	No	Yes
Interpretable	Partial	Limited	Partial	Limited

B. Scalability and Dimensionality Challenges

Sequence models that rely on sliding window approaches suffer from the "curse of dimensionality" when the number of distinct log or span event types exceeds typical limits [30]. As the event vocabulary grows, the model's ability to generalize degrades, and memory requirements increase polynomially.

Graph models mitigate this challenge through several mechanisms. First, soft-attention mechanisms allows the model to dynamically focus on nodes contributing most to an anomaly, effectively reducing the effective dimensionality of the problem [31]. Second, message-passing architecture of GNNs enables efficient information aggregation across graph neighborhoods without requiring explicit enumeration of all possible event sequences. These properties allow graph-based systems to scale effectively even as the number of events in a trace reaches 800–900 or more [25], [32].

2. Discussion

A. Practical Implications

The findings presented in this analysis have significant implications for practitioners designing observability systems for microservice architectures. The superior performance of graph-based approaches suggests that organizations should prioritize these methods for new deployments, particularly in systems with complex service dependencies and parallel execution patterns.

However, the choice between approaches should also consider organizational factors. Sequencebased models may remain appropriate for simpler architectures with predominantly linear call patterns, or in environments where training time constraints outweigh inference performance requirements. Additionally, the interpretability of sequence models may provide advantages in certain debugging and root cause analysis scenarios.

B. Limitations of Graph-Based Approaches

While graph-based models demonstrate clear advantages in structural awareness and detection accuracy, several practical limitations warrant consideration.

First, GNN-based methods are sensitive to incomplete or malformed trace data. In production environments, spans may be lost due to sampling policies, network partitions, or instrumentation failures, resulting in partial graphs. Unlike sequence models that degrade gracefully when elements are missing from a linear sequence, graph models may propagate errors through message-passing layers, amplifying the effect of missing nodes or edges on downstream representations [23].

Second, evolving microservice topologies introduce a cold-start challenge. When new services are deployed or existing call paths change, the graph structure diverges from patterns observed during training. Retraining or fine-tuning the model becomes necessary, and frequency of architectural changes in actively developed systems may make this impractical without automated retraining pipelines [1].

Third, training cost varies significantly across architectures. In our experiments, TraceVAE required 27,112 seconds of training time compared to 1,618 seconds for DeepTraLog and 7,584 seconds for DeepLog. The variational inference mechanism in TraceVAE, while yielding the highest detection accuracy, imposes a substantial computational overhead during training that must be weighed against the marginal F1 improvement over DeepTraLog (0.896 vs. 0.868).

Fourth, the explainability of GNN-based anomaly detection remains limited. While methods such as GNNExplainer [33] offer post-hoc interpretation of node-level predictions, generating humanreadable explanations for why a particular trace was flagged as anomalous is an open challenge. For operational teams, understanding the root cause behind an alert is as important as the detection itself [37].

Finally, deployment complexity is higher for graph-based systems. Constructing graph representations from raw traces at serving time requires additional preprocessing step that introduces latency and engineering effort compared to simpler tokenization required by sequence models.

C. Future Research Directions

Several promising research directions emerge from this analysis. First, hybrid architectures that combine the temporal modeling strengths of recurrent networks with the structural awareness of GNNs warrant further investigation. Second, the application of more advanced GNN variants, such as Graph Attention Networks (GATs) and Graph Transformers, to microservice anomaly detection remains underexplored. Third, the development of explainable anomaly detection methods that can identify not just that an anomaly occurred, but specifically which service interactions contributed to the anomaly, represents an important direction for operational utility [33]. Additionally, TraceGra [35] represents an existing mixed-methods strategy combining GNN and LSTM for simultaneous topology and temporal

pattern capture. Cross-system transfer learning and meta-learning approaches [42] offer a promising direction to reduce cold-start issues for new deployments. Recent comprehensive surveys [38], [39], [40] highlight the need for standardized benchmarks to enable fair cross-method comparison. Real-time distributed tracing anomaly detection in cloud environments [41] is emerging as an operational requirement. The precision-recall trade-off observed in our experiments – where GNNs favor precision while LSTMs favor recall – warrants further investigation into ensemble or cascading approaches.

7. Conclusion

This paper has compared sequence-based and graph-based deep learning approaches for anomaly detection in microservice traces through both a structured literature analysis and original experiments on the TrainTicket benchmark.

The experimental results confirm that graph-based models capture structural properties of distributed traces more effectively than sequential alternatives. In our unified evaluation, TraceVAE achieved the highest F1-score of 0.896, followed by DeepTraLog at 0.868, while the LSTM-based DeepLog reached 0.758. These findings are consistent in direction with results reported in prior independent studies, though absolute F1 values differ due to variations in data splits, preprocessing, and evaluation conditions – illustrating precisely why unified benchmarking is necessary.

A notable finding is the scale of inference efficiency gains. DeepTraLog processed the test set 237 times faster than DeepLog on identical hardware, substantially exceeding the approximately 40× speedup reported in cross-study comparisons. At the same time, our experiments reveal that graphbased models exhibit higher precision but lower recall than the LSTM baseline, suggesting that sequence models may be more sensitive to anomalies at the cost of more false positives.

These observations suggest that the choice between approaches depends on the operational context. For systems where precision is prioritized and structural fidelity matters – such as production environments with stable topologies and complete instrumentation – graph-based models offer measurable advantages. For rapidly evolving systems where high recall is critical and computational resources are constrained, sequence-based models may remain a pragmatic alternative.

Future work should investigate hybrid architectures that combine temporal and structural modeling, the application of Graph Transformers to trace analysis, and methods for generating interpretable explanations from GNN-based detectors.

Funding

The study was funded by the Science Committee of the Ministry of Science and Higher Education of the Republic of Kazakhstan, target grant # BR28713313.

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

Conceptualization, Y.Z. and A.Z.; Methodology, Y.Z., A.Z. and S.M; Software, Y.Z.; Validation, Y.Z., A.Z., A.A. and S.M; Formal Analysis, Y.Z., A.Z., A.A.. and S.M; Investigation, Y.Z., A.Z. and A.A; Resources, Y.Z. and A.Z.; Data Curation, Y.Z., A.Z. and S.M; Writing – Original Draft Preparation, Y.Z. and A.Z.; Writing – Review & Editing, Y.Z. and A.Z.; Visualization, Y.Z. and A.Z.; Supervision, A.Z.; Project Administration, A.Z.; Funding Acquisition, A.Z.

References

1. X. Zhou, X. Peng, T. Xie, J. Sun, C. Ji, W. Li, and D. Ding, "Fault analysis and debugging of microservice systems: Industrial survey, benchmark system, and empirical study," *IEEE Trans. Softw. Eng.*, vol. 47, no. 2, pp. 243–260, 2021.

2. Y. Gan, Y. Zhang, D. Cheng, A. Shetty, P. Rathi, N. Katarki, A. Bruno, J. Hu, B. Ritchken, B. Jackson, K. Hu, M. Pancholi, Y. He, P. Clancy, C. Colen, F. Wen, C. Leung, S. Wang, L. Zaruvinsky, M. Espber, R. Lin, Z. Liu, J. Padilla, and C. Delimitrou, "Seer: Leveraging Big Data to Navigate the Complexity of Performance Debugging in Cloud Microservices," in *Proc. ASPLOS*, 2019, pp. 19–33.

3. L. Wu, J. Tordsson, E. Elmroth, and O. Kao, "MicroRCA: Root Cause Localization of Performance Issues in Microservices," in *Proc. IEEE/IFIP NOMS*, 2020, pp. 1–9.

4. S. Newman, *Building Microservices: Designing Fine-Grained Systems*. O'Reilly Media, 2015.

5. A. Brandón, M. Solé, A. Huélamo, D. Solans, M. S. Pérez, and V. Muntés-Mulero, "Graph-based root cause analysis for service-oriented and microservice architectures," *J. Syst. Softw.*, vol. 159, 2020, Art. no. 110432.

6. D. Liu, C. He, X. Peng, F. Lin, C. Zhang, S. Gong, Z. Li, J. Ou, and Z. Wu, "MicroHECL: HighEfficient Root Cause Localization in Large-Scale Microservice Systems," *arXiv:2103.01782*, 2021.

7. B. H. Sigelman, L. A. Barroso, M. Burrows, P. Stephenson, M. Plakal, D. Beaver, S. Jaspan, and C. Shanbhag, "Dapper, a Large-Scale Distributed Systems Tracing Infrastructure," Google, Tech. Rep., 2010.

8. P. Mace and R. Fonseca, "Universal Context Propagation for Distributed System Instrumentation," in *Proc. EuroSys*, 2018, pp. 8:1–8:18.

9. P. Liu, H. Xu, Q. Ouyang, R. Jiao, Z. Chen, S. Zhang, J. Yang, L. Mo, J. Zeng, W. Xue, and D. Pei, "Unsupervised Detection of Microservice Trace Anomalies through Service-Level Deep Bayesian Networks," in *Proc. ISSRE*, 2020, pp. 48–58.

10. S. He, J. Zhu, P. He, and M. R. Lyu, "Loghub: A Large Collection of System Log Datasets towards Automated Log Analytics," *arXiv:2008.06448*, 2020.

11. N. Dragoni, S. Giallorenzo, A. L. Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina, "Microservices: yesterday, today, and tomorrow," in *Present and Ulterior Software Engineering*, 2017, pp. 195–216.

12. J. Thönes, "Microservices," *IEEE Softw.*, vol. 32, no. 1, pp. 116–116, 2015.

13. M. Fowler and J. Lewis, "Microservices: A definition of this new architectural term," *martinfowler.com*, 2014. [Online]. Available: https://martinfowler.com/articles/microservices. html

14. R. Sambasivan, R. Fonseca, I. Shafer, and G. R. Ganger, "So, you want to trace your distributed system? Key design insights from years of practical experience," Carnegie Mellon University, Tech. Rep., 2014.

15. J. Kaldor, J. Mace, M. Bejda, E. Gao, W. Kuropatwa, J. O'Neill, K. W. Ong, B. Schaller, P. Shan, B. Viscomi, V. Venkataraman, K. Veeraraghavan, and Y. J. Song, "Canopy: An End-to-End Performance Tracing And Analysis System," in *Proc. SOSP*, 2017, pp. 34–50.

16. M. Du, F. Li, G. Zheng, and V. Srikumar, "DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning," in *Proc. ACM CCS*, 2017, pp. 1285–1298.

17. S. Nedelkoski, J. Cardoso, and O. Kao, "Anomaly Detection from System Tracing Data Using Multimodal Deep Learning," in *Proc. IEEE CLOUD*, 2019, pp. 179–186.

18. S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

19. W. Meng, Y. Liu, Y. Zhu, S. Zhang, D. Pei, Y. Liu, Y. Chen, R. Zhang, S. Tao, P. Sun, and R. Zhou, "LogAnomaly: Unsupervised Detection of Sequential and Quantitative Anomalies in Unstructured Logs," in *Proc. IJCAI*, 2019, pp. 4739–4745.

20. A. Brown, A. Tuor, B. Hutchinson, and N. Nichols, "Recurrent Neural Network Attention Mechanisms for Interpretable System Log Anomaly Detection," in *Proc. AMLC*, 2018, pp. 1–8.

21. Z. Chen, J. Liu, W. Gu, Y. Su, and M. R. Lyu, "Experience Report: Deep Learning-based System Log Analysis for Anomaly Detection," *arXiv:2107.05908*, 2021.

22. C. Zhang, X. Peng, C. Sha, K. Zhang, Z. Fu, X. Wu, Q. Lin, and D. Zhang, "DeepTraLog: Tracelog combined microservice anomaly detection through graph-based deep learning," in *Proc. ICSE*, 2022, pp. 623–634.

23. H. X. Nguyen, S. Zhu, and M. Liu, "A Survey on Graph Neural Networks for MicroserviceBased Cloud Applications," *Sensors*, vol. 22, no. 23, 2022, Art. no. 9492.

24. Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated Graph Sequence Neural Networks," *arXiv:1511.05493*, 2015.

25. Z. Xie, H. Chen, R. Cheng, Q. Lu, Y. Wang, L. Wang, H. Chen, and M. Yang, "Unsupervised Anomaly Detection on Microservice Traces through Graph VAE," in *Proc. WWW*, 2023, pp. 2874–2884.

26. X. Zhou, X. Peng, T. Xie, J. Sun, C. Xu, C. Ji, and W. Zhao, "Poster: Benchmarking microservice systems for software engineering research," in *Proc. ICSE-C*, 2018, pp. 323–324.

27. D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *arXiv:1312.6114*, 2013.

28. T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," *arXiv:1609.02907*, 2016.

29. J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural Message Passing for Quantum Chemistry," in *Proc. ICML*, 2017, pp. 1263–1272.

30. R. Bellman, "Dynamic Programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.

31. P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," *arXiv:1710.10903*, 2017.

32. W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive Representation Learning on Large Graphs," in *Proc. NeurIPS*, 2017, pp. 1024–1034.

33. Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "GNNExplainer: Generating Explanations for Graph Neural Networks," in *Proc. NeurIPS*, 2019, pp. 9244–9255.

34. C. Lee, T. Yang, Z. Chen, Y. Su, and M. R. Lyu, "Eadro: An End-to-End Troubleshooting Framework for Microservices on Multi-source Data," in *Proc. ICSE*, 2023, pp. 1750–1762.

35. J. Chen, F. Liu, G. Zhong, J. Jiang, D. Xu, Z. Tan, and S. Shi, "TraceGra: A trace-based anomaly detection for microservice using graph deep learning," *Computer Communications*, vol. 204, pp. 109–117, 2023.

36. S. Zhang, P. Jin, Z. Lin, Y. Sun, B. Zhang, S. Xia, Z. Li, Z. Zhong, M. Ma, W. Jin, Z. Zhang, D. Zhu, and D. Pei, "Robust Failure Diagnosis of Microservice System Through Multimodal Data," *IEEE Trans. Serv. Comput.*, vol. 16, no. 5, pp. 3537–3550, 2023.

37. G. Yu, P. Chen, Y. Li, H. Chen, X. Li, and Z. Zheng, "Nezha: Interpretable Fine-Grained Root Causes Analysis for Microservices on Multi-modal Observability Data," in *Proc. ESEC/FSE*, 2023, pp. 553–565.

38. Z. Z. Darban, G. I. Webb, S. Pan, C. C. Aggarwal, and M. Salehi, “Deep Learning for Time Series Anomaly Detection: A Survey,” *ACM Comput. Surv.*, vol. 57, no. 1, pp. 1–42, 2024.

39. T. Wang and G. Qi, “A Comprehensive Survey on Root Cause Analysis in (Micro) Services: Methodologies, Challenges, and Trends,” *arXiv:2408.00803*, 2024.

40. S. Zhang, S. Xia, W. Fan, B. Shi, X. Xiong, Z. Zhong, M. Ma, Y. Sun, and D. Pei, “Failure Diagnosis in Microservice Systems: A Comprehensive Survey and Analysis,” *arXiv:2407.01710*, 2024.

41. M. Raeiszadeh, A. Ebrahimzadeh, A. Saleem, R. Glitho, J. Eker, and R. A. F. Mini, “Real-Time Anomaly Detection Using Distributed Tracing in Microservice Cloud Applications,” in *Proc. IEEE CloudNet*, 2023, pp. 1–7.

42. Y. Wang, M. Mäntylä, J. Nyyssölä, K. Ping, and L. Wang, “Cross-System Software Log-based Anomaly Detection Using Meta-Learning,” in *Proc. SANER*, 2025, pp. 1–12.

Information about the authors:

Yesset Zhussupov is a PhD student in Software Engineering at L.N. Gumilyov Eurasian National University. He received his MSc in Software Engineering from the same university in 2022. Mr. Zhussupov serves as Chief Technology Officer at an AFSA-licensed brokerage organization, where he architects systems integrating big data and machine learning technologies. His research focuses on neural network applications in finance and economics, with particular emphasis on market forecasting and predictive modeling. His recent work includes the paper “Study of the Possibilities of Using Deep Artificial Intelligence in Forecasting the Green Paper Market in Kazakhstan.” In 2015, he received the Intel Excellence Award for his work applying neural networks to computer vision–based emotion recognition (Almaty, Kazakhstan, e-mail: yesset.zhussupov@gmail.com. ORCID iD: 0009-0001-0946-8094).

*PhD Dr. Ainur Zhumadillayeva is an Associate Professor of Associate Professor of Computer and Software Engineering at L.N. Gumilyov Eurasian National University. She received her PhD degree in the specialty *05.13.18 – Mathematical Modeling, Numerical Methods, and Software Packages* from the Institute of Mathematics of the Ministry of Education and Science of the Republic of Kazakhstan (Almaty) in 2010. Ainur Zhumadillayeva is the author and coauthor of educational publications and scientific monographs and has published more than 60 scientific and educational-methodical works in Kazakhstan and abroad, including publications indexed in international databases such as Scopus and Clarivate Analytics. Her research interests include mathematical modeling, numerical methods, machine learning, and artificial neural networks. She is a Member of the Institute of Electrical and Electronics Engineers (IEEE) (Astana, Kazakhstan. ORCID iD: 0000-0003-1042-0415).*

Shona Shinassylov is a Researcher and Lecturer at the Faculty of Information Technology and Artificial Intelligence, Al-Farabi Kazakh National University. He received his Master’s degree in Information Technology from Al-Farabi Kazakh National University and completed his doctoral studies in 2025. His research interests include digital twins, artificial intelligence, machine learning, deep reinforcement learning, smart buildings, the Internet of Things (IoT), energy management systems, and intelligent decision-support systems. His current research focuses on the design and development of virtual prototypes of smart building energy management systems based on digital twin technologies, machine learning methods, and multi-objective reinforcement learning (Almaty, Kazakhstan ORCID iD: 0009-0002-5491-7255).

Ainur Amirtayeva is a Master’s student in the dual-degree program between Al-Farabi Kazakh National University (KazNU) and Northwestern Polytechnical University (NWPU). She received her Bachelor’s degree in Information Security Systems from Al-Farabi Kazakh National University. Her academic and professional interests include cybersecurity, artificial intelligence, machine learning, deep learning, neural networks, parallel and distributed programming, phishing detection, computer vision, network traffic analysis, intrusion detection systems, and the application of AI in healthcare. Her research focuses on the development of intelligent information security systems, the application of neural networks, and the use of parallel computing techniques to improve machine learning algorithms (Almaty, Kazakhstan, e-mail: amirtayevaainur@gmail.com. ORCID iD: 0009-0008-4579-287X).

Submission received: 17 January, 2026
Revised: 28 April, 2026
Accepted: 4 May, 2026