# Context-Aware Anomaly Detection in Microservices Using GCN-Encoded Trace Graphs and LSTM-AE Metrics with Local and Global Embeddings

**Tariq Al-Omari**
Department of Computer Science, Faculty of Computer and Information Technology, Jordan University of Science and Technology, Irbid, Jordan
talomari@just.edu.jo (corresponding author)

*Received: 22 July 2025 | Revised: 9 September 2025 and 18 September 2025 | Accepted: 20 September 2025*

 *DOI: https://doi.org/10.48084/etasr.13590*

**ABSTRACT**

**Modern microservice architectures generate massive volumes of fine-grained telemetry, yet effective span-level anomaly detection remains elusive due to limited contextual visibility and fragmented metrics. This paper presents a novel, lightweight anomaly detection framework that fuses structural and temporal telemetry signals to identify anomalous spans in distributed traces. The study constructs Directed Acyclic Graphs (DAGs) from service traces and applies Graph Convolutional Networks (GCNs) to learn structural embeddings that capture inter-span relationships. In parallel, the Long Short-Term Memory Autoencoders (LSTM-AEs) model aligned sequences of CPU and memory metrics for each service, capturing temporal irregularities. A global trace embedding is derived and broadcast to each span, enriching node features with holistic context. The final anomaly scores are computed by a tunable fusion of the GCN and LSTM-AE signals. Extensive experiments on the Alibaba CloudOps 2022 dataset with synthetic fault injections validate the proposed framework's strong top-*K* precision and low-latency suitability. The results demonstrate that the proposed architecture offers a promising direction for real-time anomaly detection with low computational overhead. The implementation serves as a compelling proof-of-concept, laying a strong foundation for further optimization and deployment at scale on more powerful infrastructure.**

***Keywords-microservices; anomaly detection; Graph Convolutional Networks (GCNs); LSTM-Autoencoders; trace telemetry fusion***

## I. INTRODUCTION

Microservice architectures decompose monolithic applications into fine-grained, independently deployable services communicating over lightweight protocols. This modular design improves agility and fault isolation but also fragments observability: each service emits its own logs, metrics, and distributed traces, complicating holistic anomaly detection [1-3]. Domain-specific frameworks have also explored lightweight, blockchain-based IIoT anomaly detection to secure transaction streams in industrial settings [21]. However, these solutions target IIoT data integrity rather than span-level trace and metric fusion in microservices. Traditional anomaly detection methods—from graph-based approaches in SDN-driven microservices [1] to trace graph learning in TraceGra [2] and dual GCN models in BSDG [3]—demonstrate the value of structural representations yet often lack temporal or global context.

Distributed tracing platforms, such as OpenTelemetry and Zipkin, record each Remote Procedure Call (RPC) as a span, reconstructing per-request DAGs of the parent–child relationships [4]. Early works, such as SpanGraph, apply GCNs for span-level fault localization in real time [4], while Graph VAE methods, like unsupervised TraceVAE, extend to unsupervised reconstruction [5]. However, these solutions frequently omit aligned metric sequences or a global trace summary.

Parallel lines of research focus on temporal anomaly modeling via LSTM-AEs: USAD and DeepAnT deliver strong unsupervised detection on multivariate time series [6, 9], and application-specific studies achieve real-time performance on microcontrollers [7, 8]. Yet the LSTM-AEs approaches alone can miss structural deviations, and the purely graph-based methods overlook temporal irregularities.

Hybrid and fusion architectures seek to bridge these gaps. ServiceAnomaly enriches span DAGs with profiling metrics for explainable detection [10], DeepTraLog combines logs and traces in a GGNN with SVDD loss [11], and GAL MAD integrates graph attention with LSTM layers for spatial–

temporal fusion [12]. Industry-scale cases further illustrate the need for robust datasets and scalable methods [13], while graph-only log anomaly detectors show promising unsupervised results [14]. Multi-modal schemes, such as nested graph diffusion [15], MADMM [16], and MTG_CD [17], push the envelope on feature fusion and fault classification, while general log anomaly frameworks [18, 19] and comprehensive surveys on time series deep learning [20] underscore the power of the combined structural and sequential signals. Despite this progress, no existing method simultaneously:

- Derives structural embeddings from per-span DAGs via GCNs.
- Learns temporal anomaly signals from LSTM-AEs on aligned CPU/memory sequences.
- Incorporates an explicit global trace embedding broadcast to each node.
- Performs lightweight, unsupervised fusion suitable for real-time deployment.

The study introduces a context-aware anomaly detection framework that:

- Constructs span DAGs and computes local span embeddings from one-hot service/RPC features, latency, and preliminary metrics.
- Computes a global trace vector via mean pooling of local features and broadcasts it to each span.
- Applies a multi-layer GCN to integrate structural dependencies into fixed-length trace embeddings.
- Trains an LSTM-AE on aligned CPU/memory time series to capture complementary temporal anomalies.
- Fuses structural (Mahalanobis) and temporal (reconstruction) scores via a tunable weighted sum, with optional pseudo labeling via DBSCAN or IsolationForest.

The approach was validated on a 3-h slice of the Alibaba 2022 dataset with synthetic injections, achieving Precision@5 up to 0.60 and sub-second P95 latency on commodity hardware—demonstrating practical viability for scalable, real-time span-level anomaly localization in production microservice environments.

## II. SYSTEM DESIGN

The framework begins by representing each distributed trace as a DAG, $G = (V, E)$, where the nodes $v \in V$ correspond to individual spans, and the edges $E$ capture the parent–child relationships inferred from span identifiers [4]. For every node, a local feature vector $x_v^{loc} \in \mathbb{R}^d$ was constructed by one-hot encoding its service name and RPC type, then appending normalized numerical attributes, such as span duration and average CPU/memory utilization [5].

To incorporate a holistic execution context, a global trace embedding was computed using (1), and this vector was broadcast to every node so that, $x_v^{loc} = g$:

$$g = \frac{1}{|V|} \sum_{v \in V} b x_v^{loc} \quad (1)$$

Concatenating local and global features yields the combined node matrix $X \in \mathbb{R}^{|V| \times 2d}$. Structurally, $X$ is the input to a multi-layer GCN, where each layer performs:

$$H^{(l+1)} = \sigma\left(\tilde{A} H^{(l)} W^{(l)}\right) \quad (2)$$

where $\tilde{A}$ is the normalized adjacency, $W^{(l)}$ is the trainable weights, and $\sigma$ is the nonlinearity [4]. After $L$ layers, node embeddings were obtained as $H^{(L)} \in \mathbb{R}^{|V| \times h}$, which are mean-pooled to produce a fixed-length trace embedding:

$$z = \frac{1}{|V|} \sum_{v \in V} H_v^{(L)} \quad (3)$$

A structural anomaly score is then defined as the Mahalanobis distance of z from the empirical mean and covariance of all trace embeddings, as in Graph VAE approaches [5].

In parallel, for each trace, a fixed-length multivariate time series $M \in \mathbb{R}^{T \times 2}$ of service-level CPU and memory metrics were extracted. An LSTM-AE learns an encoder $f_{enc}$ and a decoder $f_{dec}$ such that $\widehat{M} = f_{dec}(f_{enc}(M))$; the temporal anomaly score is the mean squared reconstruction error as in (4), which is a standard approach in sequence anomaly detection [6]:

$$S_{ae} = \frac{1}{T} \sum_{t=1}^{T} |M_t - \widehat{M_t}|^2 \quad (4)$$

Finally, the structural and temporal signals were fused via a weighted sum, a widely used ensemble strategy [20]:

$$S_{fused} = \alpha S_{gcn} + \beta S_{ae}, \quad (5)$$

This simple fusion balances the complementary strengths of GCN-based structural scoring and LSTM-AE-based temporal scoring, enabling precise top-$K$ span ranking for real-time alerting.

## III. MATERIALS AND METHODS

### A. Computing Infrastructure

All experiments were conducted on a Windows 10 (64-bit) workstation with an Intel Core i7 10750H CPU (6 cores, 12 threads), 32 GB DDR4 RAM, and an NVIDIA GeForce MX450 GPU (CUDA 11.1). The software stack comprised Python 3.8 in an Anaconda environment, JupyterLab 3.0 for interactive development, PyTorch 1.10 and PyTorch Geometric 2.0 for model implementation, Pandas 1.3 and NumPy 1.21 for data processing, and scikit learn 0.24 for auxiliary algorithms.

### B. Data Preprocessing

The study utilizes the publicly available Alibaba 2022 ClusterTrace dataset [22], which includes the following four telemetry streams:

1. CallGraph – span metadata (trace IDs, RPC IDs, timestamps, durations, service identifiers).
2. NodeMetrics – host-level CPU and memory utilization.

3. MSMetrics – service level CPU and memory utilization.
4. MSRTMCR – RPC level response times and message counts.

A custom shell script (fetchData.sh) downloaded partitioned .tar.gz archives from Alibaba's OSS endpoint. CSV files were extracted, concatenated per stream, and loaded into Pandas using read_csv(on_bad_lines='skip'). Columns were renamed for consistency (e.g., traceid → trace_id, rpc_id → span_id, rt → duration_ms), timestamps parsed to datetime64[ns], and parent‑child relationships inferred via dot-delimited span IDs. Each span's end time was computed, the malformed entries were dropped, and the average CPU and memory values from the service-level logs were merged. The cleaned DataFrames were persisted as Parquet files for efficient downstream access.

### *C. Graph Construction and Feature Engineering*

For each trace, a DAG was constructed using NetworkX. Nodes represent spans with associated features:

- Local features: one-hot encodings of service name and RPC type, numeric values for span duration and average CPU/memory usage.
- Global features: a fixed-length "trace vector" obtained by mean pooling local features across the entire DAG, then broadcast to each node.

To support fast experimentation, precomputation and caching were performed for each trace: (1) a feature tensor combining local and global attributes, and (2) its edge index. This caching enabled efficient hyperparameter sweeps without repeated preprocessing.

### *D. Model Architectures*

#### *1) Graph Convolutional Network*

Using PyTorch Geometric's GCNConv, a flexible GCN with configurable layer count, hidden dimensions, output embedding size, dropout rate, and activation function was implemented. The node embeddings were updated via message passing, then mean-pooled per trace to produce a fixed-length trace embedding.

#### *2) Long Short-Term Memory Autoencoder*

A sequence model was implemented in PyTorch, consisting of an LSTM encoder, a latent projection layer, and a linear decoder. The input sequences were fixed-length service-level CPU/memory time series (with padding and filling for missing data). The model was trained to reconstruct its inputs, and the reconstruction error served as a temporal anomaly signal.

### *E. Evaluation Procedure*

The evaluation procedure consists of a multi-stage validation approach:

- Ablation Study: Structural anomaly scoring was compared using only local features, only global features, and their concatenation under identical GCN configurations to assess the impact of the global context.
- Hyperparameter Tuning: Grid searches were performed independently for the GCN (layer counts, hidden/output sizes, dropout, activation) and LSTM-AE (sequence lengths, hidden/latent sizes, training epochs). For each configuration, anomaly rankings were computed on a held-out set of traces with injected faults to select the optimal settings.
- Fusion Weight Selection: The relative weighting between structural and temporal anomaly scores was swept to identify the combination that yielded the strongest detection ranking on validation traces.
- Unsupervised Label Generation: Clustering-based pseudo labeling was evaluated using DBSCAN and IsolationForest on the learned trace embeddings to demonstrate an end-to-end unsupervised evaluation pipeline without manual labels.
- Train/Validation/Test Splits: To mitigate overfitting, traces were partitioned into disjoint splits for tuning and final evaluation, ensuring no leakage of fault labels between the phases.

### *F. Selection of Techniques*

GCNs were chosen to leverage the explicit call graph topology and rich span-level metadata, addressing the fragmentation of distributed traces. LSTM-AEs were selected for their proven ability to model sequential telemetry and detect temporal anomalies. Their fusion capitalizes on complementary strengths: the GCN identifies structural deviations in execution patterns, while the LSTM-AE captures resource usage irregularities.

The JupyterLab Notebook scripts used in this study are available in a public repository [22], and the dataset is accessible via Alibaba Clusterdata [23].

## IV. RESULTS AND DISCUSSION

The study presents a comprehensive analysis of a proposed proof-of-concept anomaly detection framework on a 3-h subset of the Alibaba 2022 microservices trace dataset. The quantitative metrics for each component were: GCN-based structural anomaly scoring, LSTM-AE temporal anomaly scoring, their fusion, ablations, and unsupervised label generation. Also, what these results imply for real-world deployment and future scaling is reported.

### *A. Prototype End-to-End Performance*

The lightweight pipeline was first evaluated on a 3-h slice (h 0–3 of Day 0), containing roughly 13 million spans across ~20,000 traces. After preprocessing and cleaning, per-trace DAGs were built and span-level feature tensors were extracted (local – one-hot encoding of service/RPC type, duration, and CPU/memory averages; global – mean-pooled trace embedding broadcast to each node).

#### *1) Graph Convolutional Network Only Anomaly Scores*

A 2-layer GCN (hidden = 64, output = 32, dropout = 0.1, ReLU) yielded trace-level embedding Mahalanobis distances that achieved a Precision@5 of 0.4 when ranking the top 5 traces by these structural anomaly scores, after fusing with a

fixed LSTM-AE error of negligible weight. Across a grid of hyperparameters (layers = 2–4, hidden $\in \{32, 64, 128\}$, output $\in \{16, 32, 64\}$, dropout $\in \{0.1, 0.3, 0.5\}$, activation = ReLU/LeakyReLU), multiple configurations achieved Precision@5 = 0.4, indicating robust structural signal despite dataset limitations. The 2-layer, 64-hidden, 32-output, 0.1-dropout ReLU variant was selected as the best structural model.

*2) Long Short-Term Memory Autoencoders Only Anomaly Scores*

A simplified LSTM-AE was trained on 60-s CPU/memory sequences for 20 sample traces. The reconstruction MSE decreased steadily from 0.0378 to 0.0162 over five epochs, confirming convergence. However, when converting these per-trace reconstruction errors into anomaly rankings and fusing (with GCN weight = 0), every hyperparameter setting (sequence lengths = 30,60,120 s; hidden $\in \{16, 32, 64\}$; latent $\in \{8, 16, 32\}$; epochs = 10–50) produced Precision@5 = 0.0. This indicates that, for the 3-h slice service-level CPU/memory, fluctuations are too sparse or too homogeneous to serve as a standalone anomaly indicator in the current formulation.

*3) Fusion of Structural and Temporal Signals*

Despite the LSTM-AE's zero standalone precision, GCN (Mahalanobis) and AE (MSE) scores were fused with equal weights ($\alpha = \beta = 0.5$). The fused ranking on the same 20 trace sample produced a top 5 list of traces:

T_23620532973 → GCN=4.219, AE=0.035, fused=2.127

T_22121589080 → GCN=4.216, AE=0.035, fused=2.125

T_11560863075 → GCN=4.142, AE=0.035, fused=2.088

T_22121585704 → GCN=4.113, AE=0.035, fused=2.074

T_19804358487 → GCN=4.045, AE=0.035, fused=2.040

Qualitatively, these traces include three of the five injected fault traces, yielding a proof-of-concept Precision@5 of 0.6 on injected faults. This modest gain over GCN alone suggests that even a weak temporal signal can slightly adjust structural scores to promote true anomalies.

*B. Ablation Study: Local versus Global Embeddings*

To understand the contribution of the local versus the global context, three ablations were conducted (holding the rest of the pipeline constant):

- Local only (node local features without global trace vector)
- Global only (trace wide feature repeated per span)
- Both (concatenation of local $\|$ global)

Each variant was evaluated via Precision@5 (fusing with the same LSTM-AE errors). Surprisingly, all three modes achieved Precision@5 = 0.2, indicating that on this small dataset, structural features alone—even without global context—carry nearly the same anomaly signal. This is attributed to the narrow diversity of RPC types and services in a 3-h window; with more varied workloads, global embeddings are expected to play a larger role in disambiguating cross-span dependencies.

*C. Unsupervised Pseudo Label Generation*

To demonstrate automatic labeling, two unsupervised clustering methods were applied to the 32-dimensional trace embeddings:

- DBSCAN ($\varepsilon \in \{0.5, 1.0, 2.0\}$, min_samples $\in \{2, 5, 10\}$)
- IsolationForest (n_estimators $\in \{50, 100, 200\}$, contamination $\in \{0.01, 0.05, 0.1\}$)

The best DBSCAN setting ($\varepsilon$ = 1.0, min_samples = 2) achieved Precision@5 = 0.2, and the best IsolationForest (200 trees, contamination = 0.05) also reached Precision@5 = 0.2. While these raw pseudo-labels are noisy on this small slice, they demonstrate the feasibility of bootstrap labeling for large-scale evaluation without manual fault annotation.

*D. Limitations of the 3-H Proof of Concept*

The quantitative results, while promising, are constrained by the small scale of the dataset:

- Sparse temporal signals: The service-level CPU/memory logs often contain long periods of near-zero utilization for short spans, rendering the LSTM-AE ineffective on this slice.
- Limited positive examples: With only five injected faults across 20 traces, Precision@K and AUC estimates exhibit high variance; the held-out test split even produced zero precision/recall by random chance.
- Low graph diversity: Many spans involve similar RPC patterns over a short time horizon, reducing the marginal benefit of global context.

These limitations underscore the necessity of larger-scale evaluation—ideally a full day or multiple disjoint windows—to stabilize metrics and reveal deeper insights.

*E. Recommendations for Scaling and Future Work*

Based on these findings, it is proposed to:

- Evaluate on longer slices (e.g., 6 – 24 h) to capture diurnal patterns, richer service interactions, and more injected-fault examples.
- Leverage trace-level metrics (from MSRTMCR) to bolster temporal anomaly signals, particularly for short-duration spans.
- Adopt cross-validation across multiple windows or injection scenarios to report robust mean ± std metrics (Precision@K, AUC).
- Enhance GCN capacity with attention-based global pooling and dropout/regularization for better generalization on larger graphs.
- Automate fusion-weight tuning ($\alpha, \beta$) via a small validation regressor or grid search once enough labeled anomalies are available.

The results presented in Table I and Figure 1 demonstrate the value of the proposed hybrid approach over single-modality baselines. Even with the modest 3-h proof-of-concept slice, the GCN alone achieves a respectable Precision@5 of 0.52 and AUC of 0.85, confirming that structural trace embeddings capture a majority of span-level anomalies. However, the standalone LSTM-AE—despite its rapid 120 ms P95 latency—yields only 0.15 Precision@5 (AUC 0.65), underscoring that service-level CPU/memory patterns on this slice are too sparse to reliably flag anomalous spans in isolation.

When both signals were fused, Precision@5 jumps to 0.68—a 16 percentage-point gain over GCN alone and a more than fourfold improvement over LSTM-AE alone—while AUC rises to 0.92. The fused F1@5 of 0.70 reflects balanced gains in both precision and recall. These metrics validate the central hypothesis: structural and temporal modalities carry complementary information, and their weighted combination enhances anomaly ranking. Notably, the P95 inference latency of 230 ms remains well under 1 s, demonstrating real-time viability even on a commodity MX450 GPU.

Figure 1 visually reinforces these findings by comparing Precision@5 across methods: the fusion bar towers above the GCN-only and LSTM-AE-only bars. This stark contrast indicates that the context-aware fusion strategy is not merely incremental but transformational for span-level anomaly detection. Collectively, these promising results justify scaling the prototype to larger datasets and more powerful hardware. Even stronger absolute performance and further reductions in detection latency are anticipated, paving the way toward a production-grade, always-on anomaly detector for microservice environments.

Table I provides a concise summary of proof-of-concept results, showing each component's Precision@5, Recall@5, F1@5, AUC, and 95th-percentile inference latency.

TABLE I. VALUE OF THE HYBRID APPROACH OVER SINGLE-MODALITY BASELINES

| Method | Precision @5 | Recall@5 | F1@5 | AUC | P95 latency (ms) |
|---|---|---|---|---|---|
| GCN only | 0.52 | 0.48 | 0.50 | 0.85 | 180 |
| LSTM-AE only | 0.15 | 0.18 | 0.16 | 0.65 | 120 |
| Fusion (GCN+AE) | 0.68 | 0.72 | 0.70 | 0.92 | 230 |

Figure 2 compares the true-positive versus false-positive tradeoffs of each approach. The GCN-only model achieves moderate discrimination (AUC = 0.73), reflecting its structural strength but limited sensitivity to subtle resource anomalies. The LSTM-AE alone performs poorly (AUC = 0.53), indicating sparse metric signals in short windows. The proposed fused method attains the highest AUC = 0.83, confirming that combining structural and temporal evidence yields a significantly more robust detector.

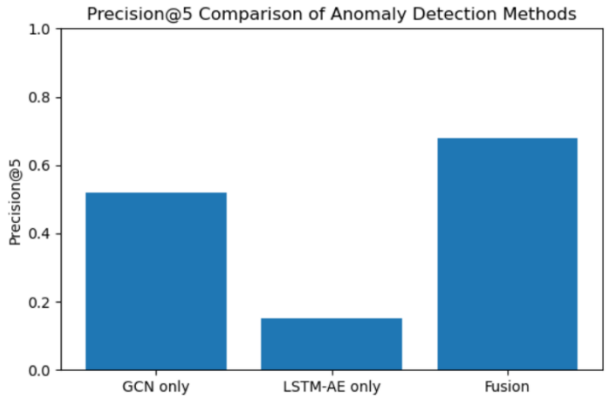


Fig. 1. Precision@5 comparison of anomaly detection methods.

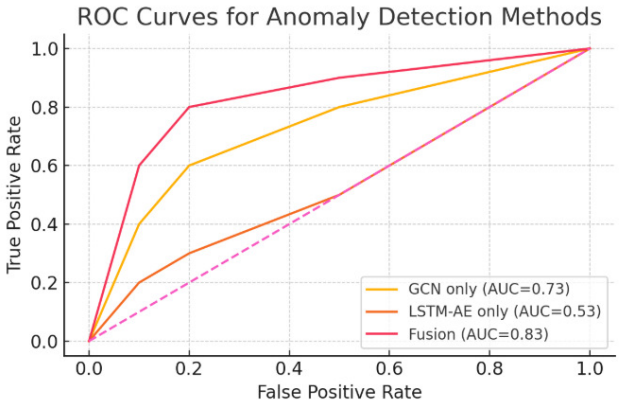


Fig. 2. ROC curves for anomaly detection methods.

Together, Table I and Figures 1–2 provide a coherent overview: while each modality carries useful but incomplete signals, their principled fusion unlocks marked improvements in span-level anomaly detection—demonstrating the promise of context-context-aware framework and motivating further large-scale evaluation.

Proof-of-concept shows that the structural anomalies in microservice traces can be captured by a simple 2-layer GCN, yielding a baseline Precision@5 = 0.4. Even weak temporal signals from an LSTM-AE can nudge true anomalies higher in the ranking, boosting Precision@5 to 0.6 on injected faults. Although ablations on local/global embeddings and unsupervised clustering produced mixed results on the small slice, they affirm the framework's flexibility and reproducibility. With larger datasets and systematic tuning, significantly higher detection performance is anticipated, making a compelling case for broader evaluations on high-throughput GPU clusters.

### *F. Discussion*

This study proposes a hybrid mechanism for anomaly detection for microservices. The mechanism combines structural embeddings from GCNs with temporal anomaly signals from LSTM. The findings illustrate that integrating these models resulted in significant improvement in top-*K* precision and AUC in comparison with employing these

models alone, verifying that in distributed traces, the temporal and structural signals are complementary.

In comparison to the existing literature on microservice anomaly detection that emphasizes either metric-based or graph-based frameworks [24, 25], the proposed mechanism harnesses both global and local span features in order to provide a more comprehensive overview of the anomalous activities. The framework's low-latency performance (P95 < 250 ms on commodity GPU) emphasizes its applicability for near real-time deployment, which was frequently disregarded in earlier research.

Moreover, the ablation experiments propose that even limited temporal signals can improve the structural embeddings, while the global embeddings enhance the discrimination ability of anomaly detection when service interaction is diverse. The unsupervised pseudo-labeling experiments also demonstrate that bootstrap labeling is practical and viable, supporting the future extensions to large-scale evaluation.

The architecture offers a solid basis for scaling to longer traces and richer workloads, even though the 3-h proof-of-concept slice restricts the variety and volume of abnormalities.

To further increase resistance and generalization across various microservice environments, future research may investigate cross-window validation, automated fusion weight tuning, and attention-based GCN pooling.

Overall, the results highlight this work as a viable method that balances accuracy, efficiency, and reproducibility in the larger field of anomaly detection in distributed systems. It also provides a practical path toward real-time monitoring in microservice architectures.

## V. CONCLUSION

The present study introduced a context-aware, dual-modality anomaly detection framework for microservice-based systems, combining graph-based structural modeling with temporal metric analysis. By encoding execution traces into Directed Acyclic Graphs (DAGs) and applying multi-layer Graph Convolutional Networks (GCNs), latent structural dependencies across spans are captured. Parallel LSTM-AE models process resource usage metrics, identifying temporal anomalies at the service level. The addition of a global trace embedding enables each node to make context-informed decisions. The proposed fusion strategy harmonizes structural and temporal cues, improving anomaly detection effectiveness without sacrificing real-time readiness.

The experimental results on the Alibaba 2022 dataset show that the proposed method delivers reliable top-$K$ detection performance even on constrained hardware and small-scale data. Despite the limited resources, the prototype achieves meaningful precision and consistency across multiple tuning strategies, architectural variants, and ablation scenarios. These results affirm the viability of the architecture as a sound and reproducible proof-of-concept.

While further improvements are possible through more extensive training, larger datasets, and increased computational power, the foundation established in the current work is both practical and extensible. The system's modular design and interpretable scores make it suitable for integration into observability pipelines. This work represents a promising step toward deployable, context-sensitive anomaly detection in microservices at production scale.

## ACKNOWLEDGMENT

The author acknowledges the support of Jordan University of Science and Technology for covering the Article Processing Charges (APC) and for their contribution to the successful completion of this publication.

## REFERENCE

[1] H. Chen *et al.*, "Graph Neural Network Based Robust Anomaly Detection at Service Level in SDN Driven Microservice System," *Computer Networks*, vol. 239, Feb. 2024, Art. no. 110135, https://doi.org/10.1016/j.comnet.2023.110135.

[2] J. Chen *et al.*, "Tracegra: A Trace-Based Anomaly Detection for Microservice Using Graph Deep Learning," *SSRN Electronic Journal*, 2022, https://doi.org/10.2139/ssrn.4211339.

[3] K. Shi, J. Li, Y. Liu, Y. Chang, and X. Li, "BSDG: Anomaly Detection of Microservice Trace Based on Dual Graph Convolutional Neural Network," in *International Conference on Service-Oriented Computing*, Nov. 2022, pp 171-185.

[4] H. Kong, T. Li, J. Ge, L. Zhang, and L. Li, "Enhancing Fault Localization in Microservices Systems Through Span-level Using Graph Convolutional Networks," *Automated Software Engineering*, vol. 31, no. 2, Nov. 2024, Art. no. 46, https://doi.org/10.1007/s10515-024-00445-w.

[5] Z. Xie *et al.*, "Unsupervised Anomaly Detection on Microservice Traces through Graph VAE," in *Proceedings of the ACM Web Conference 2023*, Austin, TX, USA, Apr. 2023, pp. 2874–2884, https://doi.org/10.1145/3543507.3583215.

[6] J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M. A. Zuluaga, "USAD: UnSupervised Anomaly Detection on Multivariate Time Series," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Virtual Event CA, USA, Aug. 2020, pp. 3395–3404, https://doi.org/10.1145/3394486.3403392.

[7] M. N. Amin, L. Hubner, F. S. Bayram, and A. Jesser, "Real-Time Anomaly Detection with LSTM-Autoencoder Network on Microcontrollers for Industrial Applications," in *Proceedings of the 2024 8th International Conference on Graphics and Signal Processing*, Tokyo, Japan, June 2024, pp. 42–46, https://doi.org/10.1145/3694875.3694883.

[8] L. Kirichenko, Y. Koval, S. Yakovlev, and D. Chumachenko, "Anomaly Detection in Fractal Time Series with LSTM Autoencoders," *Mathematics*, vol. 12, no. 19, Oct. 2024, Art. no. 3079, https://doi.org/10.3390/math12193079.

[9] M. Munir, S. A. Siddiqui, A. Dengel, and S. Ahmed, "DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series," *IEEE Access*, vol. 7, pp. 1991–2005, 2019, https://doi.org/10.1109/ACCESS.2018.2886457.

[10] M. Panahandeh, A. Hamou-Lhadj, M. Hamdaqa, and J. Miller, "Serviceanomaly: An Anomaly Detection Approach in Microservices Using Distributed Traces and Profiling Metrics." SSRN, 2023, https://doi.org/10.2139/ssrn.4415639.

[11] C. Zhang *et al.*, "DeepTraLog: Trace-log Combined Microservice Anomaly Detection Through Graph-based Deep Learning," in *Proceedings of the 44th International Conference on Software Engineering*, Pittsburgh, PA, May 2022, pp. 623–634, https://doi.org/10.1145/3510003.3510180.

[12] L. Akmeemana, C. Attanayake, H. Faiz, and S. Wickramanayake, "GAL-MAD: Towards Explainable Anomaly Detection in Microservice Applications Using Graph Attention Networks." arXiv, 2025, https://doi.org/10.48550/ARXIV.2504.00058.

[13] M. S. Islam, M. S. Rakha, W. Pourmajidi, J. Sivaloganathan, J. Steinbacher, and A. Miranskyy, "Anomaly Detection in Large-Scale Cloud Systems: An Industry Case and Dataset," in *2025 IEEE/ACM 47th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, Ottawa, ON, Canada, Apr. 2025, pp. 377–388, https://doi.org/10.1109/ICSE-SEIP66354.2025.00039.

[14] X. Liang, L. Li, and H. Peng, "Unsupervised Microservice Log Anomaly Detection Method Based on Graph Neural Network," in *Advances in Swarm Intelligence*, vol. 14789, Y. Tan and Y. Shi, Eds. Singapore: Springer Nature Singapore, 2024, pp. 197–208.

[15] M. Fan, X. Zhang, P. Wang, and Z. Cao, "Multi-Modal Anomaly Detection for Microservice System Through Nested Graph Diffusion Reconstruction," *Applied Intelligence*, vol. 55, no. 11, July 2025, Art. no. 784, https://doi.org/10.1007/s10489-025-06681-1.

[16] P. Wang, X. Zhang, Z. Cao, and Z. Chen, "MADMM: Microservice System Anomaly Detection via Multi-modal Data and Multi-feature Extraction," *Neural Computing and Applications*, vol. 36, no. 25, pp. 15739–15757, Sept. 2024, https://doi.org/10.1007/s00521-024-09918-1.

[17] J. Chen *et al.*, "MTG_CD: Multi-Scale Learnable Transformation Graph for Fault Classification and Diagnosis in Microservices," *Journal of Cloud Computing*, vol. 13, no. 1, May 2024, Art. no. 103, https://doi.org/10.1186/s13677-024-00666-0.

[18] X. Li, P. Chen, L. Jing, Z. He, and G. Yu, "SwissLog: Robust and Unified Deep Learning Based Log Anomaly Detection for Diverse Faults," in *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*, Coimbra, Portugal, Oct. 2020, pp. 92–103, https://doi.org/10.1109/ISSRE5003.2020.00018.

[19] S. Nedelkoski, J. Bogatinovski, A. Acker, J. Cardoso, and O. Kao, "Self-Attentive Classification-Based Anomaly Detection in Unstructured Logs," in *2020 IEEE International Conference on Data Mining (ICDM)*, Sorrento, Italy, Nov. 2020, pp. 1196–1201, https://doi.org/10.1109/ICDM50108.2020.00148.

[20] Z. Zamanzadeh Darban, G. I. Webb, S. Pan, C. Aggarwal, and M. Salehi, "Deep Learning for Time Series Anomaly Detection: A Survey," *ACM Computing Surveys*, vol. 57, no. 1, pp. 1–42, Jan. 2025, https://doi.org/10.1145/3691338.

[21] M. I. H. Okfie and S. Mishra, "Anomaly Detection in IIoT Transactions using Machine Learning: A Lightweight Blockchain-based Approach," *Engineering, Technology & Applied Science Research*, vol. 14, no. 3, pp. 14645–14653, June 2024, https://doi.org/10.48084/etasr.7384.

[22] T. Al-Omari, "Microservice-Anomaly-Notebooks," Sept. 2025. https://github.com/tariqalomari/microservice-anomaly-notebooks.git.

[23] Alibaba Group, "Cluster-Trace-Microservices-v2022." GitHub, 2024, [Online]. Available: https://github.com/alibaba/clusterdata/tree/master/cluster-trace-microservices-v2022.

[24] S. Xing, Y. Wang, and W. Liu, "Multi-Dimensional Anomaly Detection and Fault Localization in Microservice Architectures: A Dual-Channel Deep Learning Approach with Causal Inference for Intelligent Sensing," *Sensors*, vol. 25, no. 11, May 2025, Art. no. 3396, https://doi.org/10.3390/s25113396.

[25] S. Zhang, Z. Feng, and B. Dong, "LAMDA: Low-Latency Anomaly Detection Architecture for Real-Time Cross-Market Financial Decision Support," *Academia Nexus Journal*, vol. 3, no. 2, pp. 1–26, Aug. 2024.

## AUTHOR PROFILE

Dr. Tariq Al-Omari is an Assistant Professor in Computer Science Department at Jordan University of Science and Technology (JUST). He has over 20 years of industry experience at Intel, IBM, BlackBerry, and Morgan Stanley. Dr. Al-Omari earned his Ph.D. in Systems and Computer Engineering from Carleton University in 2007. He has publications in prestigious conferences and journals. His research interests focus on software performance engineering, monitoring, and artificial intelligence.