

Architecture for the Acquisition and Preprocessing of Telemetry in Microservice Systems for Anomaly Detection Using Machine Learning Models

Henry Ricardo Barrientos Flores
Universidad Evangélica de El Salvador
(UEES), San Salvador, El Salvador
barrientoshenry342@gmail.com

Fernando Javier Herrera Mayorga
Universidad Evangélica de El Salvador
(UEES), San Salvador, El Salvador
mayorgaf11@gmail.com

Steven Orlando Rivera Meléndez
Universidad Evangélica de El Salvador
(UEES), San Salvador, El Salvador
steven.melendez001@gmail.com

Jose Alejandro Vigil Chavarría
Universidad Evangélica de El Salvador
(UEES), San Salvador, El Salvador
rocojose29@gmail.com

José Luis Ordoñez Ávila
Universidad Evangélica de El Salvador, El Salvador
Universidad Tecnológica Centroamericana (UNITEC), Honduras
jlordonez@unitec.edu; jose.ordonez@uees.sv.edu

Abstract—Microservice architectures have become the standard paradigm for cloud-native applications, but their distributed nature complicates anomaly detection because logs, metrics, and traces originate from heterogeneous sources and require consistent preprocessing before machine learning models can be applied. This paper proposes a layered architecture for the acquisition, preprocessing, feature engineering, and, when independently available, multimodal fusion of telemetry data to support anomaly detection in microservice environments. The proposed framework transforms heterogeneous observability artifacts into structured temporal feature vectors suitable for supervised and unsupervised machine learning models. The architecture was evaluated using source-specific views of public datasets derived from TraceRCA, RCAEval, TrainTicket, and Sock Shop, complemented by synthetic telemetry for controlled validation of the processing pipeline. Experimental results showed that structured telemetry preprocessing enabled F1-scores above 0.99 on TraceRCA-based scenarios, while metric-only datasets exhibited high anomaly sensitivity but lower precision due to increased false positives. The results also show why data provenance matters: derived graph-cache tensors with label-derived proxy fields must not be interpreted as evidence of a causal gain from independent multimodal fusion.

Index Terms—Microservices, anomaly detection, machine learning, observability, logs, metrics, traces, XGBoost, autoencoder, telemetry preprocessing.

I. INTRODUCTION

Microservices architecture has established itself as the dominant paradigm for cloud-native systems due to its modularity and ability to scale independently, but its distributed complexity creates challenges in terms of observability and anomaly detection [1]–[3]. Manually collecting, managing, analyzing, correlating, and interpreting large volumes of data is unfeasible [1], [3], [4]. In domains such as next-generation mobile networks or industrial infrastructures, subtle failures in the control plane or service chains can degrade service,

rendering traditional troubleshooting methods ineffective [5]–[8].

Modern observability is built around three fundamental pillars: logs, metrics, and distributed traces, which must be correlated to provide a coherent view of system behavior [1], [2]. Review studies highlight the need for observability at different levels, including system, service, and network perspectives [1], [2], [6]. At the same time, research on distributed tracing and dynamic visualization of microservice architectures emphasizes that, even with OpenTelemetry, Jaeger, or equivalent tools, developers still face changing topologies where identifying the root cause of an anomaly through dashboards and ad-hoc queries is time-consuming and error-prone [2], [3]. For this reason, architectures for acquiring and preprocessing logs and metrics are needed to feed machine learning models capable of detecting anomalies with high Accuracy and F1-score while reducing diagnosis time compared with manual approaches or rigid rule-based methods [4], [9], [10].

Traditional monitoring methods are simple to implement and easy for operators to understand, but they are insufficient when dealing with subtle anomalies and may produce many false positives when the load exhibits seasonal variations or non-stationary patterns [4], [11]–[13]. In DevOps environments, manually defining and maintaining rules for each microservice and deployment environment leads to noisy alerts, outdated thresholds, and a high cognitive load during continuous monitoring [2], [12], [14]. Furthermore, in complex systems such as cloud-native, 5G mobile networks, or big data infrastructures, many degradations originate from interactions between services and therefore cannot be captured adequately by local thresholds based on a single metric [5], [6], [8].

There is a trend toward multimodal monitoring architectures that integrate logs, metrics, and traces into distributed and highly automatable pipelines [1], [2], [6]. Work on observabil-

ity in microservices proposes stacks that unify these elements into analytics platforms where time series and call graphs are constructed [1]–[3]. In telecommunications and cloud-native service scenarios, this approach is complemented by control-plane-specific probes and proactive orchestration mechanisms, so that the acquisition layer supports both operational use cases and near-real-time anomaly detection [5], [6], [14]. Some proposals also move instrumentation closer to the kernel, combining network and system telemetry with lower overhead and enabling adaptive sensors for ML models that consume high-frequency event streams [15]–[18].

To leverage logs and metrics using machine learning, careful preprocessing is required to reduce noise, synchronize sources, and generate structured representations [9], [11]. For logs, several studies apply parsing techniques to extract patterns and separate constants from parameters, followed by semantic vectorization so that each event can be used for classification or anomaly detection [11], [13]. At the trace and response-time level, statistical thresholds based on latency distributions can filter low-latency traffic considered normal and focus training on regions where spikes and heavy tails are observed [10], [16], [19]. Other works define service call graphs with enriched attributes and construct features that capture both intra-service behavior and causal dependencies between microservices [9], [11], [19], [20].

On top of these acquisition and preprocessing architectures, supervised classifiers based on tree ensembles such as XGBoost have shown strong performance when combined with class balancing and hyperparameter optimization, including F1-scores close to 99.9% on TraceRCA-based scenarios [10]. Other studies apply robust principal component analysis, clustering, asynchronous federated learning, graph-based approaches, and explainable security models to separate normal behavior from anomalous components or to reason about invocation chains [8], [9], [11], [13], [15], [16], [20]. Based on this background, the objective of this work is to design and evaluate a compact, reproducible architecture that integrates telemetry acquisition, preprocessing, temporal alignment, multimodal feature fusion, and machine learning-based anomaly detection.

II. METHODOLOGY

The proposed solution is an offline experimental framework for anomaly detection in cloud-native microservice systems. Its purpose is to transform heterogeneous observability artifacts into consistent machine-learning inputs while keeping the origin, modality, label source, and experimental view of each dataset explicit. The methodology follows six explicit layers: (1) input acquisition, (2) preprocessing, (3) feature engineering, (4) training preparation, (5) model training, and (6) output evaluation. The flow between layers is sequential, while modality-specific log, metric, and trace operations may run in parallel before feature fusion. This organization reflects current observability literature, where logs, metrics, and traces must be acquired and aligned before higher-level anomaly detection can be reliable [1], [2], [4].

A. Input and Dataset Registration (Layer 1)

The input layer receives public datasets and local research artifacts associated with cloud-native anomaly detection. The framework does not assume that all sources contain the same observability modalities. Instead, each source is registered according to the available structure. TraceRCA-main is used as an explicit-label CSV view with trace, latency, and anomaly fields. TraceRCA all dataset and RCAEval are treated as recursive service-fault CSV views where service folders, metric files, and injection metadata define the experimental context. Sock Shop is used as a metric-only scenario-aware view with chaos metadata, which makes it useful for evaluating alert behavior under deployment-level perturbations; the local view does not contain independent raw distributed logs and traces. TrainTicket is represented through derived graph-cache tensors that combine service-dependency and telemetry-derived information. This source is valuable for studying graph-based feature separability, but it is not described as a complete raw logs, metrics, and traces corpus.

The registration step stores the physical data source, experimental view, telemetry modality, label provenance, and independence status before model training. This prevents a graph-cache tensor, a raw CSV file, and a metric-only chaos experiment from being interpreted as the same type of evidence. It also aligns the paper with the way recent microservice studies separate telemetry collection, root-cause context, and model evaluation [9], [11], [15].

B. Preprocessing (Layer 2)

The preprocessing layer converts each artifact into clean and comparable records. The operations include timestamp normalization, duplicate removal, missing-value handling, metric resampling, service-name normalization, and alignment between telemetry windows and available anomaly labels. For trace-oriented sources, span duration, trace depth, fanout proxies, failed-span indicators, and latency summaries are extracted. For metric-oriented sources, CPU, memory, network, response-time, and request-rate columns are standardized into a shared schema. For graph-cache sources, tensor objects are flattened into tabular vectors while preserving the original graph-derived labels.

For logs, the prototype does not use TF-IDF, Word2Vec, or a language model. The function used for ϕ_L lowercases each message, replaces standalone numbers with `<num>`, replaces trace identifiers with `<trace>`, and collapses repeated whitespace. Each temporal window then uses log count, error and warning counts, unique normalized-template count, and error-log ratio. Thus, ϕ_L is a frequency-and-severity representation rather than a semantic NLP embedding. For TrainTicket, serialized `.val` graph samples contain node-level trace, log, and metric tensors; numeric arrays are flattened and summarized with counts, means, maxima, p95/p99 proxies, and depth/fanout proxies. The target is obtained from the graph label, so this is a derived graph-cache representation and not a raw three-pillar telemetry corpus. In addition, the current adapter derives auxiliary log/trace proxy fields from that same

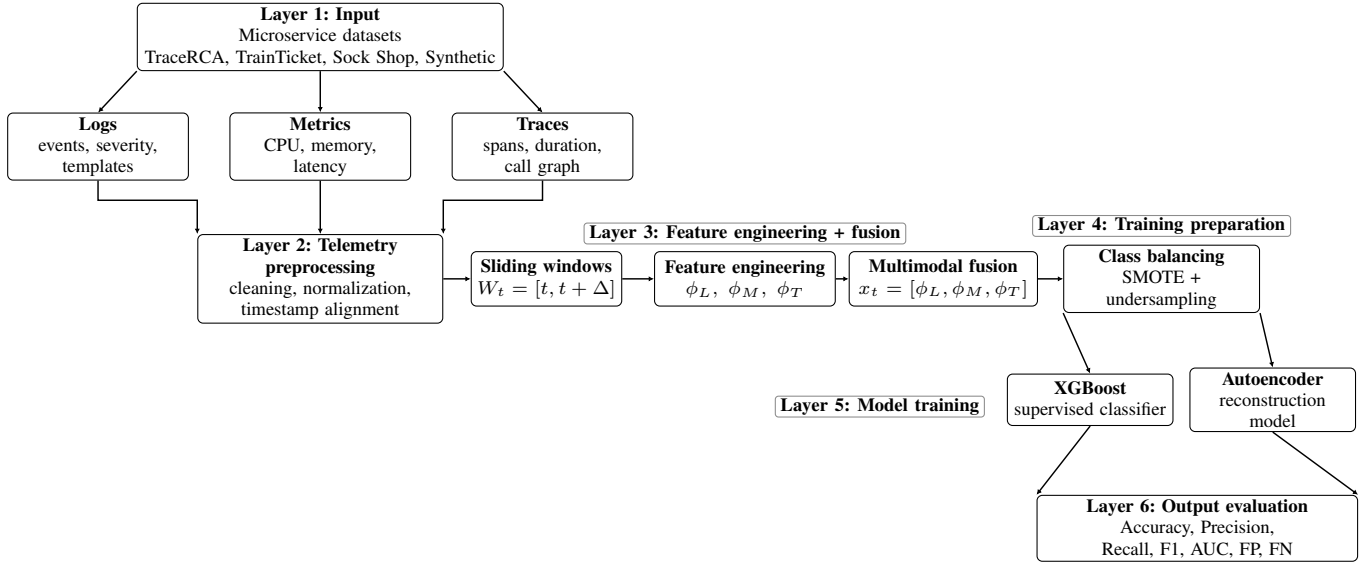


Fig. 1. Six-layer vector flowchart of the proposed anomaly-detection framework. Layers 1–6 are explicitly labeled in the diagram. The input datasets feed parallel log, metric, and trace branches; the resulting features pass through temporal windows, fusion, class balancing, model training, and evaluation.

label, including fault-class messages or severity and trace-error indicators. These fields are not independent observations and may leak the target; consequently, the TrainTicket result is evidence about separability in the derived cache, not about a clean gain from independent multimodal fusion.

The preprocessing design follows two principles. First, feature construction is source-aware: a dataset that lacks local raw logs is not artificially described as log-based. Second, label construction is conservative: explicit anomaly columns and injection windows are preferred, while derived labels are documented through the corresponding source metadata. These decisions are important because cloud-native failures often propagate through service dependencies, and an apparently local metric spike may be caused by an upstream latency degradation or downstream saturation [9], [16], [19].

C. Feature Engineering (Layer 3)

Let L , M , and T denote the log, metric, and trace telemetry streams, respectively. For each temporal window

$$W_t = [t, t + \Delta],$$

the framework extracts modality-specific feature representations when those modalities are available. The log feature extractor is denoted as $\phi_L(W_t)$, the metric feature extractor as $\phi_M(W_t)$, and the trace feature extractor as $\phi_T(W_t)$. The final multimodal representation, applicable only when the constituent inputs are independently available, is defined as

$$x_t = [\phi_L(W_t), \phi_M(W_t), \phi_T(W_t)]. \quad (1)$$

The target variable is binary:

$$y_t \in \{0, 1\}, \quad (2)$$

where 0 represents normal behavior and 1 represents anomalous behavior. This representation allows the system to transform heterogeneous telemetry into a tabular machine learning dataset, making it suitable for XGBoost classification and autoencoder-based reconstruction analysis.

D. Training Preparation (Layer 4)

Training preparation is performed after feature construction to prevent data leakage. Splitting is applied before scaling or balancing. Chronological training, validation, and test partitions are preferred because operational telemetry is time ordered. If the chronological test partition contains only one class, a deterministic stratified split is used so that Precision, Recall, F1-score, and AUC-ROC remain mathematically valid. Scalers are fitted only on X_{train} , and class balancing is applied only to the training partition. This is consistent with supervised anomaly-detection studies that combine structured preprocessing with class balancing and hyperparameter optimization [10], [12], [13].

E. Model Training (Layer 5)

The model layer evaluates three families of detectors. XGBoost is the main supervised model because it performs well on structured tabular telemetry and is directly comparable with the ADMXGB approach reported for TraceRCA-style data [10]. Random Forest is included as an ensemble baseline to verify whether the separability of the engineered features depends specifically on gradient boosting. The autoencoder is used as a reconstruction-based anomaly detector, which provides an unsupervised-style comparison when labels are limited or when reconstruction error is a useful operational signal.

For the autoencoder, the anomaly score is defined as:

$$s(\mathbf{x}) = \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2. \quad (3)$$

A sample is classified as anomalous when:

$$s(\mathbf{x}) > \tau, \quad (4)$$

where τ is selected from validation reconstruction errors.

F. Output Evaluation (Layer 6)

The evaluation layer reports Accuracy, Precision, Recall, F1-score, AUC-ROC, true negatives, false positives, false negatives, and true positives. False positives represent normal windows incorrectly escalated as anomalies, while false negatives represent anomalous windows missed by the detector. For this reason, the confusion matrix is treated as an output-layer artifact rather than a preprocessing artifact.

III. RESULTS AND DISCUSSION

The experimental results show that the proposed preprocessing and feature-engineering architecture can generate highly separable representations for explicit-label TraceRCA-style data, while also exposing the difficulty of transferring metric-only behavior across different chaos scenarios. All values reported in Fig. 2 and Fig. 3 are means from the extended ten-seed suite (42, 123, 2026, 7, 13, 21, 99, 314, 2718, and 9001). The strongest CSV-wide performance was obtained on TraceRCA-main, where Random Forest reached an F1-score of 0.9969 and AUC-ROC of 1.0000, while XGBoost reached an F1-score of 0.9967 and AUC-ROC of 1.0000. These values are close to the ADMXGB published TraceRCA reference, which reports approximately 99.96% Accuracy and F1-score using XGBoost with preprocessing, data balancing, and hyperparameter tuning [10].

The TraceRCA-main result indicates that the engineered latency, trace, and statistical window features provide a strong supervised signal when explicit anomaly labels are available. The difference between Random Forest and XGBoost is small, which suggests that the feature representation itself is informative and not dependent on a single classifier. The autoencoder also performs strongly on this view, reaching an F1-score of 0.9803 and AUC-ROC of 0.9974. Its lower Precision compared with the tree-based models indicates that reconstruction error is sensitive to abnormal regions, but it may mark additional normal windows as suspicious when the normal distribution is broad.

The TraceRCA service-fault and RCAEval results are also strong, with XGBoost F1-scores of 0.9998 and 0.9993, respectively. These views rely on service-fault metric windows rather than the same explicit-label CSV used in TraceRCA-main. Their high performance supports the usefulness of service-level aggregation when fault metadata is available. The TrainTicket graph-cache result reached F1-score and AUC-ROC of 1.0000 for the evaluated object labels; however, this is derived, limited supporting evidence because the graph-cache label and serialized tensors are not equivalent to an independent raw multimodal corpus. In particular, label-derived log/trace proxy fields in the current adapter mean that this result must not be interpreted as evidence of a causal improvement from fusing independent modalities. This

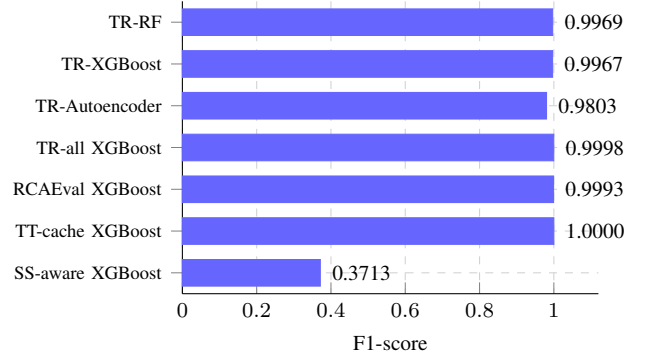


Fig. 2. Ten-seed mean F1-scores for the seven evaluated view/model combinations. TR-RF, TR-XGBoost, and TR-Autoencoder use the explicit-label TraceRCA-main CSV; TR-all and RCAEval are recursive service-fault CSV windows; TT-cache is derived TrainTicket graph-cache evidence; and SS-aware is the Sock Shop metrics-plus-injection-metadata transfer view. Values near 1.0 indicate strong classification on the corresponding view, whereas the lower SS-aware value exposes the limitation of metric-only transfer across chaos scenarios.

agrees with prior work that emphasizes invocation chains, dependency graphs, and causal structures for anomaly analysis in microservice systems [9], [11], [19], [20].

A. Operational Interpretation

The Sock Shop scenario-aware experiment produced a different behavior. XGBoost reached high Recall (0.9667), but Precision was 0.2299 and AUC-ROC was approximately 0.5000. This means that the model detected most anomalous windows, but it also escalated many normal windows as anomalies. In operational terms, this behavior is useful as an early-warning detector but insufficient as a precise alerting mechanism. It confirms that metric-only telemetry with chaos metadata can reveal abnormal behavior but may not contain enough context to separate normal load variation from fault-induced degradation. This limitation supports the need for logs, traces, service dependencies, and richer causal features, as described in observability and multimodal root-cause localization research [1], [2], [4].

The confusion matrix is essential because Accuracy alone can be misleading in imbalanced telemetry. True negatives (TN) represent normal windows correctly classified as normal. False positives (FP) represent normal windows incorrectly escalated as anomalies. False negatives (FN) represent anomalous windows missed by the detector. True positives (TP) represent anomalous windows correctly detected. A high FP count increases alert fatigue and operator distrust, while a high FN count allows incidents to remain undetected. Therefore, Precision, Recall, F1-score, AUC-ROC, FP, and FN must be analyzed together.

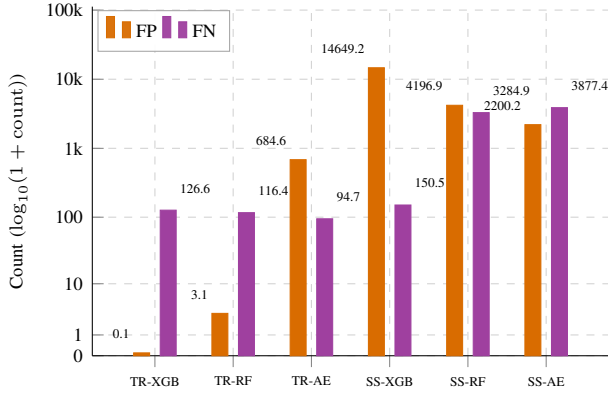


Fig. 3. Ten-seed mean false-positive (FP) and false-negative (FN) counts for the TraceRCA-main and Sock Shop scenario-aware views. TR denotes TraceRCA-main, SS denotes Sock Shop, and XGB, RF, and AE denote XGBoost, Random Forest, and Autoencoder, respectively. Bar heights use the pseudo-logarithmic transformation $\log_{10}(1 + \text{count})$, while the tick labels report the original counts; this preserves a zero origin and keeps values spanning several orders of magnitude visible. FP are normal windows incorrectly flagged as anomalous, FN are anomalous windows missed by the detector, and each decimal value is the mean across the ten seeds.

Fig. 3 summarizes the false-positive and false-negative behavior across the evaluated datasets. TraceRCA-main produced very few false positives for both Random Forest and XGBoost, whereas the Autoencoder generated a larger number of false alarms. In contrast, the Sock Shop dataset exhibited substantially higher false-positive rates, explaining why high Recall did not translate into high Precision. On TraceRCA-main, XGBoost generated almost no false positives and a limited number of false negatives, while Random Forest produced a similar balance with slightly higher false positives and fewer missed anomalies. The autoencoder produced a stronger false-positive burden than the tree-based models, although its Recall remained high. On Sock Shop, the metric-only scenario transfer generated a large false-positive count, especially for XGBoost, which explains why its high Recall did not translate into a high F1-score. This finding is consistent with the general literature: stronger observability does not come only from collecting more values, but from correlating metrics with traces, logs, service topology, and fault context [2], [3], [14].

B. Comparison With Related Work

The closest numeric comparison is ADMXGB, which uses XGBoost, class balancing, and hyperparameter tuning for anomaly classification in TraceRCA and reports approximately 99.96% Accuracy and F1-score [10]. The proposed framework reaches a comparable level on TraceRCA-main with both Random Forest and XGBoost, while also adding an explicit preprocessing and dataset-registration layer that distinguishes CSV-wide, service-fault, graph-cache, and scenario-aware metric views. This is important because microservice datasets often differ in how they expose labels, service identifiers, and telemetry modalities.

Other studies provide methodological support rather than direct one-to-one numeric comparison. RPCA-based methods emphasize invocation-chain analysis and robust decomposi-

tion for microservice anomaly detection [9]. Federated and multimodal approaches highlight the relevance of distributed training, span causal graphs, and cross-source telemetry fusion [11]. DeLag focuses on latency degradation and shows that service-based systems require attention to response-time distributions rather than only binary status indicators [19]. eBPF-based and adaptive sensing approaches also support the idea that anomaly detection can improve when low-level telemetry and service context are combined [16]. Together, these works justify the proposed layered architecture: acquisition and pre-processing must be designed before the classifier is selected.

Although the prototype is evaluated offline, the same pipeline could be adapted to streaming by using eBPF or OpenTelemetry probes for acquisition, Kafka for event transport, and Flink for event-time windows, watermarks, and late-event handling. The log, metric, and trace branches could be processed in parallel and joined by service and timestamp before inference. Operational latency would include collection, queueing, window completion, feature construction, inference, and alert delivery; therefore, model inference time alone is not an end-to-end latency measure. A future streaming evaluation should report throughput and p50/p95 end-to-end latency, while treating eBPF/Kafka/Flink as an implementation path rather than as components already implemented [11], [16], [17].

Overall, the results indicate that the architecture is effective when the dataset includes explicit labels or well-defined fault metadata, and that its weakest setting is metric-only transfer across different chaos scenarios. This is a useful result for cloud-native monitoring because it separates two goals that are often conflated: detecting that something abnormal is happening and identifying it with enough precision to support reliable alerting. The proposed pipeline performs strongly in the first setting when labels are explicit and features are aligned, but the Sock Shop result shows that precise alerting requires richer telemetry than metrics alone. Similar integration requirements appear in IoT-enabled systems, where sensor acquisition, machine learning inference, and connected control must operate within a coordinated architecture [21]. At the same time, connected telemetry increases the attack surface: machine learning models have been used to identify malicious network activity in IoT environments [22], while post-quantum cryptographic approaches are emerging as potential mechanisms for protecting communications in these systems [23].

IV. CONCLUSION

This paper presented a layered architecture for acquiring, preprocessing, and integrating heterogeneous telemetry data to support anomaly detection in cloud-native microservice systems. The proposed framework transforms logs, metrics, and traces into structured feature representations suitable for machine learning models, providing a reproducible pipeline for telemetry-driven anomaly detection.

Experimental results demonstrated that the proposed approach achieved excellent performance on TraceRCA-based datasets, with F1-scores above 0.99, while the Sock Shop

evaluation highlighted the limitations of metric-only telemetry due to increased false positives. These findings emphasize the importance of source-aware telemetry preprocessing and, where independent modalities are available, multimodal data integration for reliable anomaly detection in microservice environments. The TrainTicket graph-cache result is not evidence of a causal multimodal improvement because its auxiliary fields include label-derived proxies. Future work will evaluate independent raw logs, metrics, and traces and incorporate explainable AI techniques to improve service-level diagnosis and root-cause localization.

REFERENCES

- [1] U. Faseeha, H. J. Syed, F. Samad, S. Zehra, and H. Ahmed, "Observability in microservices: An in-depth exploration of frameworks, challenges, and deployment paradigms," *IEEE Access*, vol. 13, pp. 72011–72039, 2025.
- [2] J. Kosińska, B. Baliś, M. Konieczny, M. Malawski, and S. Zieliński, "Toward the observability of cloud-native applications: The overview of the state-of-the-art," *IEEE Access*, vol. 11, pp. 73036–73052, 2023.
- [3] M. E. Gortney, P. E. Harris, T. Cerny, A. Al Maruf, M. Bures, D. Taibi, and P. Tisnovsky, "Visualizing microservice architecture in the dynamic perspective: A systematic mapping study," *IEEE Access*, vol. 10, pp. 119999–120012, 2022.
- [4] D. F. Pedroso, L. Almeida, L. E. G. Pulcinelli, W. A. A. Aisawa, I. Dutra, and S. M. Bruschi, "Anomaly detection and root cause analysis in cloud-native environments using large language models and bayesian networks," *IEEE Access*, vol. 13, pp. 77550–77564, 2025.
- [5] G. Prince and R. Prabhavathi Neelakandan, "Ai-driven analysis and mitigation of control-plane signaling anomalies in next-generation mobile networks," *IEEE Access*, vol. 14, pp. 11129–11148, 2026.
- [6] R. Chowdhury, C. Talhi, H. Ould-Slimane, and A. Mourad, "Proactive and intelligent monitoring and orchestration of cloud-native ip multimedia subsystem," *IEEE Open Journal of the Communications Society*, vol. 5, pp. 139–155, 2024.
- [7] Y. Liu, D. Lan, Z. Pang, M. Karlsson, and S. Gong, "Performance evaluation of containerization in edge-cloud computing stacks for industrial applications: A client perspective," *IEEE Open Journal of the Industrial Electronics Society*, vol. 2, pp. 153–168, 2021.
- [8] Y. Guo and G. Tang, "Intelligent analysis and dynamic security of network traffic in context of big data," *Journal of Cyber Security and Mobility*, vol. 13, no. 5, pp. 823–841, 2024.
- [9] M. Jin, A. Lv, Y. Zhu, Z. Wen, Y. Zhong, Z. Zhao, J. Wu, H. Li, H. He, and F. Chen, "An anomaly detection algorithm for microservice architecture based on robust principal component analysis," *IEEE Access*, vol. 8, pp. 226397–226408, 2020.
- [10] L. M. Barata, E. Lopes, P. R. M. Inácio, and M. M. Freire, "Classification of anomalies in microservices using an xgboost-based approach with data balancing and hyperparameter tuning," *IEEE Open Journal of the Computer Society*, vol. 6, pp. 1673–1685, 2025.
- [11] M. Raeiszadeh, A. Ebrahimzadeh, R. H. Glitho, J. Eker, and R. A. F. Mini, "Asynchronous real-time federated learning for anomaly detection in microservice cloud applications," *IEEE Transactions on Machine Learning in Communications and Networking*, vol. 3, pp. 176–194, 2025.
- [12] J. Ababneh, E. Y. A. Al-Nsour, A. Al-Shaikh, M. R. Al-Mousa, A. Al-Zabin, M. Asassfeh, and H. AbuAlese, "Enhancing devops continuous monitoring phase: Hybrid intrusion detection and ensemble learning system (hidels)," *IEEE Access*, vol. 14, pp. 4733–4755, 2026.
- [13] J. Kosińska and M. Tobiasz, "Detection of cluster anomalies with ml techniques," *IEEE Access*, vol. 10, pp. 110742–110753, oct 2022.
- [14] A. Mittal, "Ai-augmented devsecops pipelines for secure and scalable service-oriented architectures in cloud-native systems," in *2025 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, (Tucson, AZ, USA), IEEE, jul 2025.
- [15] J. Nobre, E. J. S. Pires, and A. Reis, "Anomaly detection in microservice-based systems," *Applied Sciences*, vol. 13, no. 13, 2023.
- [16] H. Ran, C. Hu, and Y. Zhuang, "Bridging observability and security: A graph-based arbitration and adaptive sensing approach via ebpf," *IEEE Access*, vol. 14, pp. 19934–19951, 2026.
- [17] S. Cruzes, "Telemetry and agentic ai: Foundations for optical network automation," *IEEE Access*, vol. 14, pp. 8800–8838, 2026.
- [18] R. de la Fuente, L. Radrigan, and A. S. Morales, "Enhancing predictive maintenance in mining mobile machinery through a hierarchical inference network," *IEEE Access*, vol. 13, pp. 59480–59504, 2025.
- [19] L. Traini and V. Cortellessa, "Delag: Using multi-objective optimization to enhance the detection of latency degradation patterns in service-based systems," *IEEE Transactions on Software Engineering*, vol. 49, no. 6, pp. 3554–3580, 2023.
- [20] W. Lu, I. Traoré, I. Woungang, E. Brown, M. L. Brocardo, Q. Yu, and O. L. Soh, "Xapt: Explainable anomaly-driven prediction of threat stages in apt campaigns," *IEEE Access*, vol. 13, pp. 199737–199756, 2025.
- [21] J. L. Ordóñez-Ávila, D. D. Lopez, D. Ventura, J. Molina-Paluchó, A. D. Colindres, M. E. Perdomo, K. M. R. Leiva, J. V. Fajardo, and M. Cardona, "A 3D printing soft prosthesis and an IoT architecture for controlling gestures," *Journal of Biomimetics, Biomaterials and Biomedical Engineering*, vol. 69, pp. 137–150, 2025.
- [22] A. G. Menjivar, J. L. Ordóñez-Ávila, and M. Cardona, "Intrusion detection in smart homes using K-nearest neighbors and decision trees algorithm on IoT network traffic for attack classification," in *Proceedings of the 23rd LACCEI International Multi-Conference for Engineering, Education and Technology*, pp. 1–12, 2025.
- [23] A. G. Menjivar, "A survey in lattice-based post-quantum cryptography and its appliances on IoT systems," *Innovare Revista de Ciencia y Tecnología*, vol. 14, no. 2, pp. 1–11, 2025.