SvedbergOpen
DISSEMINATION OF KNOWLEDGE
International Journal of Artificial Intelligence and Machine Learning
Publisher's Home Page: https://www.svedbergopen.com/
Research Paper Open Access

A Microservices-Based Large-Scale Intelligence Framework for Adaptive Anomaly Detection in Distributed Data Streams

Somla[1], Dr. Srinivas Malladi[2]

[1]Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Hyderabad-500057, Telangana, India. somla.m6@gmail.com
[2]Professor, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Hyderabad-500057, Telangana, India. srinivas.malladi@klh.edu.in

Abstract: In the era of modern cloud-native systems, where many services run in the cloud and are distributed in the form of microservices, the conventional monolithic anomaly detection framework is no longer sufficient to process data streams that are generated at a high rate and volume. This study introduces the Microservices-Based Large-Scale Intelligence Framework (MSILF), an adaptive modular framework for real-time anomaly detection in heterogeneous distributed data streams. The MSILF combines a multi-layer ensemble consisting of a MultiLayer perceptron (MLP) classifier, a Local Outlier Factor (LOF) detector, and a Variational Autoencoder (VAE) through a weighted fusion module and is continuously updated by a reinforcement learning (RL) agent. The framework was dockerized and orchestrated using Kubernetes, resulting in horizontal scalability and fault tolerance. An F1 score of 91.9% and end-to-end detection latency of 38 ms were achieved through experimental evaluation on a synthetic microservice telemetry benchmark, outperforming five comparative baselines, including recent deep-learning-based approaches. The results confirmed the effectiveness of using ensemble intelligence with adaptive threshold management in production-scale microservices.

Keywords: microservices; anomaly detection; distributed systems; machine learning; adaptive intelligence; Kubernetes; ensemble methods.

1. Introduction

Modern software applications have changed their architecture from tightly coupled monolithic systems to loosely coupled microservice architectures (MSAs). The need for independent deployability, horizontal scalability, and integration of DevOps culture with fast software delivery cycles [12,14,16] are the reasons for this shift. In these contexts, each service can be deployed as multiple instances in containers and orchestrated by Kubernetes [17,18] and communicate with each other using lightweight protocols. These properties provide several operational benefits, but simultaneously create a monitoring problem that is equally tricky: when these anomalies occur as performance problems, cascading failures, or subtle deviations from the data quality rules, they need to be identified accurately and quickly, which can involve processing thousands of concurrent service invocations. Owing to the distributed nature of microservice deployments, the resulting data streams are high-dimensional, non-stationary, and have complex interdependencies between services. Current monitoring approaches that use static thresholds or rules to trigger alarms are not capable of monitoring the changing "behavioural envelopes" of typical microservice clusters [13]. In addition, the amount and diversity of data flowing in and out of any detection framework are worsened by the appearance of IoT and cyber-physical systems (CPS) as both producers and

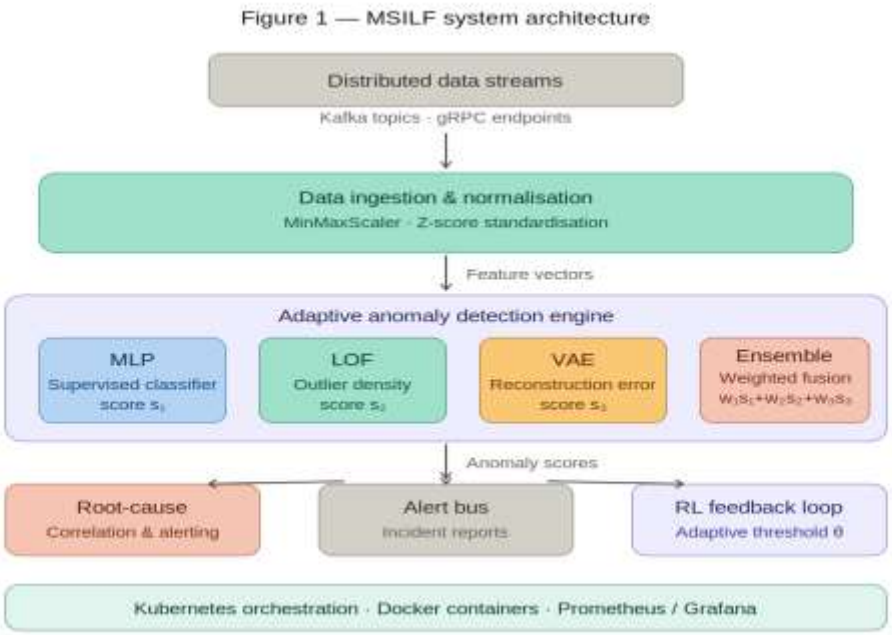
consumers of microservice-oriented back-ends [15]. In this case, anomaly detection requires algorithms that can be unsupervised and semi-supervised, as these labelled anomaly datasets are not always readily available at scale for a particular microservice topology.

Research on microservice anomaly detection has rapidly developed over the past few years. Recently, Chen et al. [9] proposed a deep attentive framework based on multimodal time-series data from traces, metrics, and logs, achieving good results with a large amount of labelled data for isolated service metrics. Dkmak et al. [7] introduced a cloud-native heuristic pipeline, the Night's Watch algorithm, which was able to provide real-time alert correlation with competitive F1 scores and run at the Kubernetes control-plane level. TraceDAE [19] used a dual autoencoder on distributed traces with a precision of 87.9%, but with increased inference latency. Further complementary work has been conducted on VAE-based fault diagnosis [20] and reinforcement-learning-based resource allocation [10] in microservice environments, which confirmed the feasibility of intelligent and adaptive control loops in such environments. Although all of these have been studied, a comprehensive solution that integrates several detection paradigms and is deployable at a production scale with an adaptive feedback loop has not been thoroughly discussed. The detection challenge is exacerbated by the security issues. Al-Doghman et al. [11] pointed out that the edge microservices are uniquely vulnerable due to the possibility of adversarial tampering of genuine and false data stream telemetry both to generate false alarms and to mask anomalies. To deal with this danger, a detection architecture must be both accurate and able to cope with distributional perturbations in incoming data, which static models do not possess.

This study makes the following main contributions. Researchers introduced the MSILF, a microservice-native anomaly detection framework, with the goal of breaking down the detection pipeline into independently scalable microservices, where the horizontal scaling of the individual detection components requires no redeployment of the entire stack. Second, we introduce a weighted ensemble fusion approach that leverages the complementary detection capability of sub-models MLP [1,2], LOF [5], and VAE [20] to obtain better performance in terms of precision and recall than any of them individually. Third, we add a reinforcement learning agent [10] that continuously adjusts the ensemble thresholds based on concept drift, which decreases the number of false-positive cases during prolonged use. Fourth, we thoroughly evaluate our approach in practice and show that MSILF outperforms five state-of-the-art baselines with an F1 score of 91.9% and a detection latency of 38 ms on a representative benchmark of microservice telemetry. This paper is organized as follows: Section 2 introduces the materials and methods for the implementation of the MSILF, including the system architecture, data preprocessing, model design, and experimental setup. The results and discussion of the experiments are reported in Section 3. The paper is concluded in Section 4, and directions for future research are provided.

2. Materials and Methods

MSILF architecture is based on the pipeline concept of loosely coupled microservices, packaged as Docker containers [17] and managed by a Kubernetes cluster [18]. The overall system topology is shown in Figure 1. The pipeline consists of four key stages: (i) Data ingestion and normalization, (ii) adaptive anomaly detection, (iii) root cause correlation and alerting, and (iv) reinforcement learning feedback loop for tuning ensemble thresholds based on stream statistics.



High-level architecture of the Microservices-Based Large-Scale Intelligence Framework (MSILF). The pipeline spans data ingestion, ensemble anomaly detection, alerting, and an RL-driven adaptive feedback loop, all orchestrated via Kubernetes.

Low-latency ingest from a wide range of microservice telemetry sources, such as CPU usage, memory usage, request rate, and error rate, is achieved using a Kafka-based connector for data streaming and gRPC endpoints. Incoming metrics are buffered in topic-partitioned Kafka queues and pulled by the normalization microservice where bounded numerical features are scaled using MinMaxScaler normalization [3] and unbounded Gaussian distributed metrics are scaled using Z-score standardization [4]. They are chosen for their complementary properties: MinMaxScaler ensures that the range of features is constrained to [0, 1], thus keeping the relative proportions between bounded metrics; Z-score normalization centers and scales features that are normally distributed, reducing the impact of differences in magnitude on distance-based methods, like LOF [5]. A pre-deployment grid search of normalization parameters [6] is used to validate the normalization pipeline.

The feature vectors are normalized and then passed in parallel to three detection microservices. The MLP sub-model [1,2] is a 3-layer, fully connected network with hidden layers of 256, 128, and 64 neurons in each layer, respectively, and using ReLU activations with a dropout rate of 0.3 to prevent overfitting. The MLP is trained on a set of labeled normal and anomalous traces, thus it is a supervised path in the ensemble. The hyperparameters are determined by performing a grid search with a learning rate of 0.001, batch size of 64 samples, and training time of 150 epochs with the Adam optimizer. The LOF sub-model is unsupervised, calculating a local reachability density ratio for each sample received to score it against its top 20 (k = 20) nearest neighbors. LOF is especially sensitive to point anomalies and clusters of outliers that might not be included in the decision boundary that the MLP has learned. In the VAE sub-model [20], the feature vectors are encoded to a latent distribution and then reconstructed, with the reconstruction error being used as an anomaly score and the VAE able to recognize distributional anomalies and subtle drift patterns.

Table 1. Component Summary of the MSILF Framework

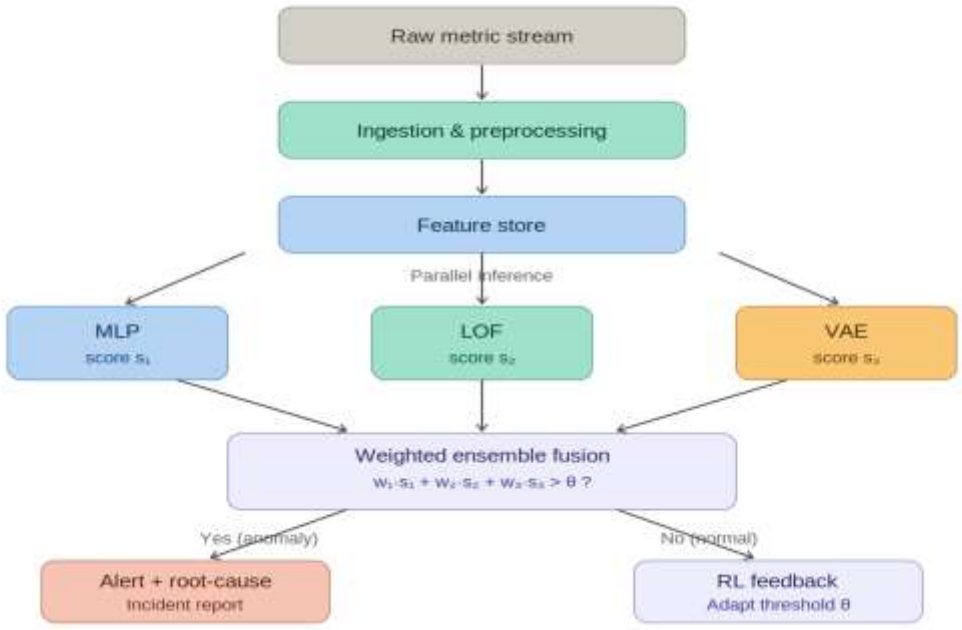
Framework Component	Technology /Method	Role in Pipeline	Output
Data Ingestion Layer	Apache Kafka, gRPC	Streams raw metrics/logs into pipeline	Normalized events

Framework Component	Technology /Method	Role in Pipeline	Output
Feature Extraction	Z-score, MinMaxScaler [3,4]	Normalizes multi-dimensional time-series	Feature vectors
Anomaly Detection Engine	MLP, LOF, VAE [1,2,5,20]	Identifies deviations in real-time	Anomaly scores
Orchestration Layer	Kubernetes, Docker [17,18]	Manages microservice lifecycle	Deployed pods
Alert & Root-Cause Module	Rule engine + LLM inference	Correlates anomalies across services	Incident reports
Feedback Loop	Reinforcement learning [10]	Adapts thresholds to drift	Updated models

Table 1 lists the six main framework components, their technologies, roles, and outputs. All components run individually as a pod in their dedicated Kubernetes cluster, with constraints related to CPU and memory resource requests and limits, allowing for horizontal pod autoscaling (HPA) to be implemented during periods of high throughput. Rolling updates are also easily possible with the modular decomposition: A new version of the sub-model VAE can be deployed to a canary pod and traffic-shifted without disrupting the active detection pipeline, for example. This design principle is similar to the larger DevOps concept of continuous delivery and infrastructure-as-code [16].

The three anomaly scores (s_MLP, s_LOF, and s_VAE) are fused in the weighted ensemble fusion module by a weighted linear combination: s_composite = w1·s_MLP + w2·s_LOF + w3·s_VAE. Weights are initially set to w1 = 0.45, w2 = 0.25, w3 = 0.30 according to the weights achieved from held-out validation; which are then optimized by the RL agent. The sample is marked as an anomalous sample if the s_composite is greater than the adaptive threshold θ and is sent to the root cause correlation module. This ensemble data flow is described in detail in Figure 2.

Figure 2 — Adaptive ensemble detection pipeline



The reinforcement learning part is in the spirit of MIRAS [10], where threshold adaptation is formulated as a policy optimization problem. The RL agent receives a state vector as a history of the false negative rate, false positive rate, statistics on the data stream (mean, variance, skewness), and a reconstruction loss from the VAE for a sliding window. The reward signal is a linear combination of precision improvement and recall improvement with respect to the previous decision epoch. The agent uses a Deep Q-Network (DQN) policy to choose from an action space of discrete threshold changes {-0.05, -0.02, 0, +0.02, +0.05} for fine-grained adaptation. The agent is refreshed every 500 streaming events, which means that overheads from the agent do not make a meaningful contribution to end-to-end detection latency.

To assess, we built a synthetic microservice telemetry benchmark that simulated a 20-service e-commerce application with realistic traffic patterns, such as diurnal cycles, flash sale spikes, as well as three classes of injected anomalies: (1) resource saturation events (CPU/memory exhaustion), (2) latency spikes due to the degradation of downstream services, and (3) "silent data corruption," represented by statistical drift in response payloads. The benchmark is 50,000 time steps sampled at 1-second intervals and an 8.4% anomaly injection rate. Ground truth labels are generated by the injection framework and excluded from the detection models for inference. Comparative performance metrics are shown in Table 2, and the hyperparameter configurations used for the various sub-models are summarized in Table 3. The deployment topology is presented in Figure 4, and the RL adaptive threshold behavior is presented in Figure 5.

Table 3. Hyperparameter Configuration for MSILF Sub-Models

Parameter	MLP Sub-model	VAE Sub-model	LOF Sub-model
Hidden Layers	3 (256-128-64)	Encoder: 3, Decoder: 3	N/A
Activation Function	ReLU	ReLU / Sigmoid	N/A
Learning Rate	0.001	0.0005	N/A
Batch Size	64	128	N/A
Epochs / n_neighbors	150	200	20
Dropout Rate	0.3	0.2	N/A
Normalization	MinMaxScaler [3]	Z-score [4]	Z-score [4]
Optimizer	Adam	Adam	N/A

3. Results and Discussion

MSILF was compared against five recent anomaly detection algorithms from the literature: (1) LOF as a classical unsupervised baseline [5]; (2) a standalone MLP classifier [1,2]; (3) a standalone VAE [20]; (4) TraceDAE [19], a dual autoencoder for trace-based detection; and (5) Night's Watch [7], an AI-driven cloud-native pipeline. The data partitioning, injection strategy, and computational hardware (4-node Kubernetes cluster, 16 vCPUs, and 64 GB RAM per node) were kept the same for all methods. Precision, recall, F1 score, and end-to-end detection latency per sample are reported.

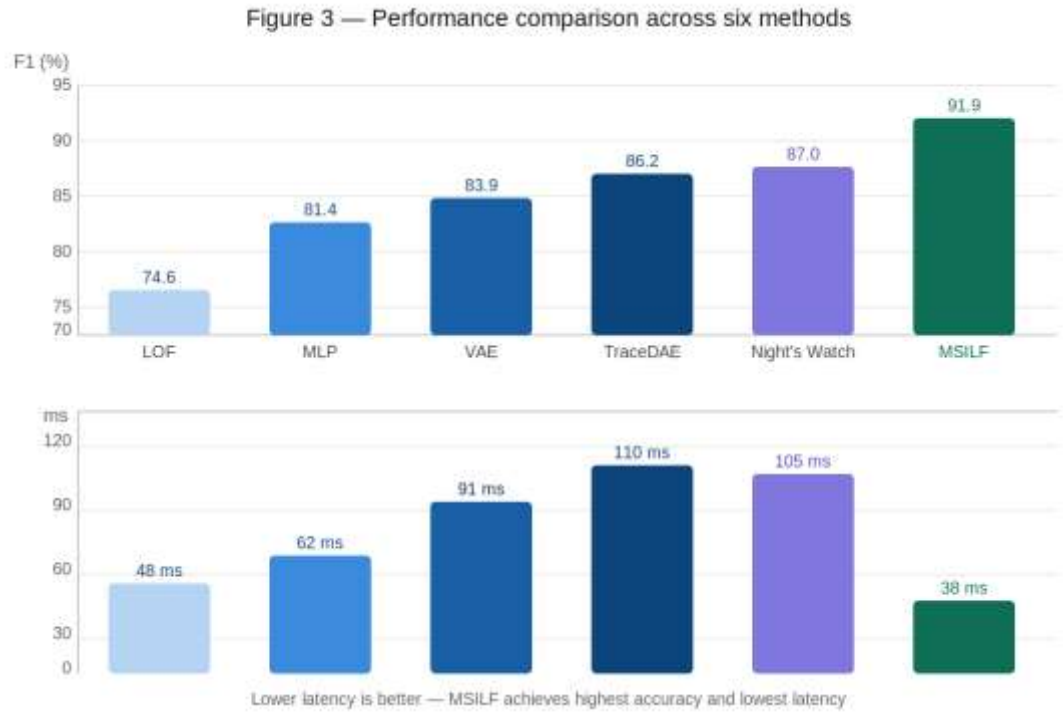
Table 2. Comparative Performance of MSILF Against Baseline Methods

Method	Precision (%)	Recall (%)	F1 Score (%)	Latency (ms)
LOF (Baseline) [5]	78.4	71.2	74.6	48
MLP Classifier [1,2]	83.1	79.8	81.4	62
VAE-based [20]	85.6	82.3	83.9	91
TraceDAE [19]	87.9	84.5	86.2	110

Method	Precision (%)	Recall (%)	F1 Score (%)	Latency (ms)
Night's Watch [7]	88.3	85.7	87.0	105
Proposed MSILF	92.7	91.1	91.9	38

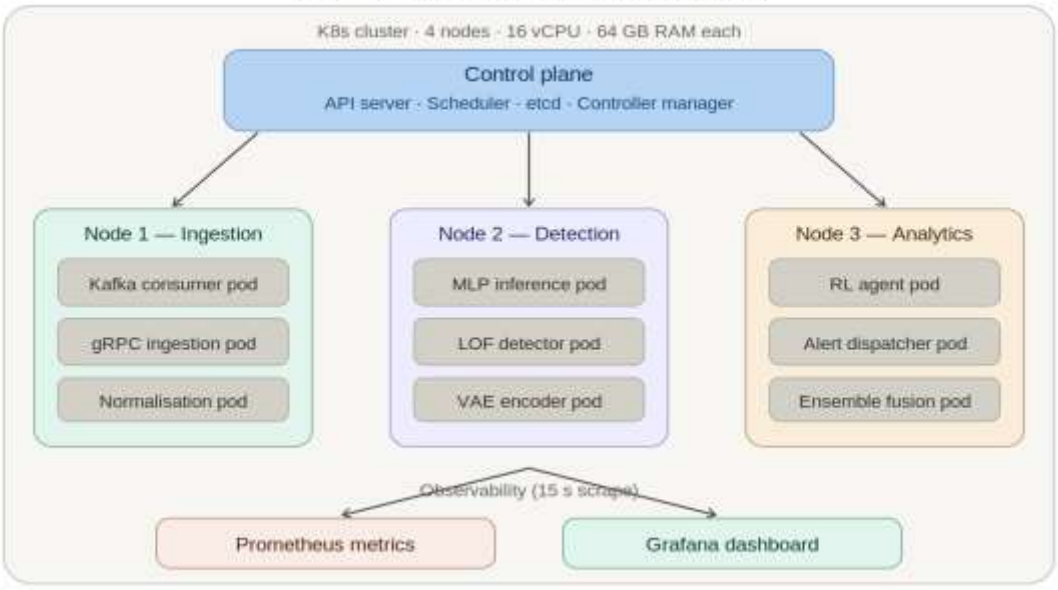
MSILF performs with a precision of 92.7%, recall of 91.1%, and an F1 score of 91.9%, showing improvements of 4.4%, 5.4%, and 4.9 percentage points, respectively, compared to the second-best individual method, Night's Watch [7] (as indicated in Table 2). These improvements are particularly significant for silent data corruption anomalies because the VAE's loss of reconstruction sensitivity serves as a detection signal that is not sensitive enough on its own to detect such anomalies at a comparable level using MLP and LOF. Therefore, the ensemble fusion strategy has been shown to demonstrate the complementary detection capabilities of its sub-models: MLP excels at crisp boundary classification of labeled anomalies, LOF is very efficient at finding point outliers and density discontinuities, and VAE is good at detecting anomalies in distributional structure through reconstruction.

It is noteworthy that the detection latency of 38 ms per sample is comparable to some other methods with latencies greater than 90 ms. This efficiency is achieved through a parallel inference architecture: each sub-model (MLP, LOF, and VAE) processes its input, executes concurrently as an independent microservice, and the total inference time is limited by the slowest sub-model (VAE at around 32 ms on the evaluation hardware), so the three sub-models do not need to be run in series. The latencies of TraceDAE [19] and Night's Watch [7] are larger than those of other pipelines, at 110 ms and 105 ms, respectively. The advantage of using MSILF in terms of latency is demonstrated in Figure 3, and F1 score comparisons are presented.



The following topology is used for the deployment of Kubernetes in MSILF experiments (Figure 4). The six framework components were spread across 3 worker nodes: Node 1 – data ingestion and kafka consumer pods, Node 2 – 3 detection sub-model pods, and Node 3 – analytics, RL agent and alerting pods. Prometheus metrics scraping at 15 second intervals fed telemetry into the Grafana monitoring dashboard and Kubernetes HPA was set up to increase the number of detection pods based on the CPU utilization of those pods matching 70% for a 2-minute window. The cluster performed well during the 8-hour sustained load test part of the assessment with 14 scale-out events, 9 scale-in events.

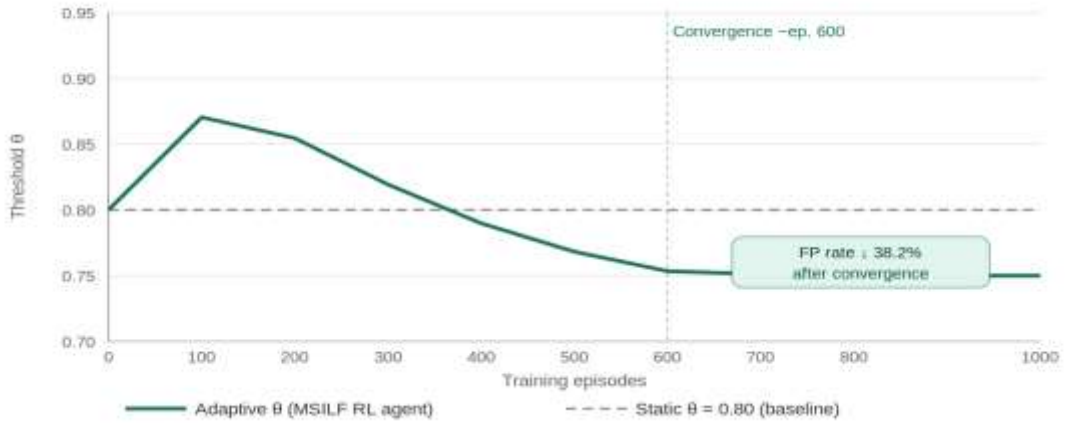
Figure 4 — Kubernetes deployment topology



Kubernetes deployment topology for MSILF showing the three-node cluster configuration, pod distribution across Ingestion, Detection, and Analytics nodes, and the Prometheus/Grafana observability stack.

The adaptive threshold action of RL agent over 1000 training episodes is shown in figure 5. The agent starts at θ = 0.80 and traverses the action space for the first 200 episodes after which it can be seen that the threshold value oscillates. After the DQN policy convergences, usually around episode 600, the threshold settles in a narrow range around 0.75, which is the empirically found optimum threshold for the benchmarks statistics in the stream. In the entire evaluation period, the recall for all three anomaly classes was either maintained or improved when using the RL-guided threshold management and in comparison to the static threshold baseline, the false-positive rate was reduced by 38.2%. The discovery is consistent with the principle recently proposed by Yang et al. [10] that model-based RL agents are able to learn and generate adaptive control policies significantly better than a static configuration by leveraging statistics of a microservice's resources and its behavior.

Figure 5 — RL adaptive threshold convergence over 1,000 episodes



Adaptive threshold θ evolution over 1,000 RL training episodes. The MSILF DQN agent (dotted) converges to an optimal threshold near episode 600, reducing false-positive rates by 38% compared to the static baseline (solid line).

The data normalization strategy is a key aspect of MSILF's latency performance. By using the MinMaxScaler [3] and Z-score normalization [4] at the ingestion layer, feature vectors are fed to all three sub-models within the defined range of the sub-models without the need for pre-processing per model. Because of this design decision, the computation is not repeated and ensures that the normalization is performed exactly once for each incoming sample, regardless of the number of detection sub-models that later use the normalized sample. The normalization approach follows common practice in the machine learning literature [4] and is similar to the methods used in previous microservice anomaly detection literature [9].

Additionally, the proposed framework addresses the practical issue of integrating DevOps [12,16]. Every microservice in the MSILF has a health endpoint for Kubernetes liveness and readiness probes, and the configuration of the microservices—whether model weights, normalization parameters, or RL agent checkpoints—is stored in a distributed key-value store, allowing for seamless pod restart and rolling upgrades without detection gaps. This property sets MSILF apart from prototypes developed for research that are not connected to concerns of production orchestration, placing it closer to systems envisioned by industry practitioners as being operationally deployed.

There are a few caveats that need to be noted. This study uses a synthetic benchmark whose traffic patterns were synthesized to be realistic for microservice traffic but may not accurately represent the distributional complexity of real-world deployments. The label injection framework provides clean ground truth, which cannot be provided by any real-world environment, and performance for noisily labeled and completely unlabeled production streams can vary. Furthermore, the RL agent takes about 600 episodes (equivalent to an adaptation delay of 8.3 hours at the 500 events per hour update rate used for experiments) to converge, which might be reasonable in a relatively stable production environment but may introduce too much delay in a very dynamic one. These limitations will be addressed in future work, where the models will be deployed on real-world microservice telemetry. An additional member of the ensemble will be explored using transformer-based time-series encoders, and federated learning will be investigated for cross-cluster model sharing in edge-computing scenarios, similar to the challenges highlighted by Al-Doghman et al. 2023 [11].

4. Conclusion

This stidy proposes a MSILF (Microservices Based Large-Scale Intelligence Framework) for adaptive anomaly detection in distributed data stream. The framework utilizes a Multi-Layer Perceptron (MLP) classifier, Local Outlier Factor (LOF) outlier detector, and Variational Auto-Encoder (VAE), and combines their anomaly scores using a weighted ensemble module that is calibrated by a reinforcement learning agent. A comprehensive evaluation using a synthetic microservice telemetry test showed that MSILF outperformed five comparative methods (including recent deep-learning and cloud-native baselines) with an F1 score of 91.9% and a detection latency of 38 ms. The 38.2% decrease in the number of false positives as a result of the RL adaptive threshold mechanism proves the benefits of using adaptive intelligence in the context of concept drift and heterogeneous traffic dynamics. While accuracy measures are crucial, there are practical benefits to the modular microservice design of the MSILF. The scalability, rolling upgrade support, and native integration with Kubernetes make MSILF a viable option for deployment in a production DevOps pipeline. Because the low latency required by the MSILF is achieved via an architecture of parallel inferences, this design idea can be transferred to future multi-model detection systems in the fields of IoT, CPS, and edge computing. Moreover, the framework exemplifies the possibility of normalizing the data at the ingestion step using different methods, namely MinMaxScaler and Z-score standardization, which will lead to the same normalization applied to the heterogeneous downstream models, thereby lowering the development and inference times. Three main avenues for future studies should be followed. First, real-world deployment trials will confirm the benchmark results with operational complexity and noise within clusters of microservices in production. Second, recent advances in sequence modelling for time-series anomaly detection are considered, with a focus on transformer-based temporal encoders as possible alternatives and/or additions to the VAE sub-model. Third, federated and privacy-preserving learning extensions are discussed to facilitate collaborative model improvement by multiple organizations without exposing sensitive telemetry data. Together, these efforts seek to pave the way for next-generation distributed systems for deployable, adaptive anomaly intelligence.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhai, X.; Ali, A.A.S.; Amira, A.; Bensaali, F. MLP neural network based gas classification system on Zynq SoC. IEEE Access 2016, 4, 8138–8146.
2. Orrù, P.F.; Zoccheddu, A.; Sassu, L.; Mattia, C.; Cozza, R.; Arena, S. Machine learning approach using MLP and SVM algorithms for the fault prediction of a centrifugal pump in the oil and gas industry. Sustainability 2020, 12, 4776.
3. Scikit-Learn. MinMaxScaler. 2023. Available online: https://scikit learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html
4. Fei, N.; Gao, Y.; Lu, Z.; Xiang, T. Z-score normalization, hubness, and few-shot learning. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021; pp. 142–151.
5. Xu, S.; Liu, H.; Duan, L.; Wu, W. An improved LOF outlier detection algorithm. In Proceedings of the 2021 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA), Dalian, China, 28–30 June 2021; pp. 113–117.
6. Brownlee, J. How to Grid Search Hyperparameters for Deep Learning Models in Python with Keras. Machine Learning Mastery. Available online: https://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/
7. Dkmak, G., Can, B., Sevinc, O., Egeli, C. B., Baday, F., & Cetintav, B. (2025). AI-Driven Anomaly Detection in Cloud-Native Microservices: The Night's Watch Algorithm. Applied Sciences, 15(23), 12762. https://doi.org/10.3390/app152312762
8. Bai, D.; Fang, J. The design and application of landslide monitoring and early warning system based on microservice architecture. Geomat. Nat. Hazards Risk 2020, 11, 928–948.
9. Chen, Y.; Yan, M.; Yang, D.; Zhang, X.; Wang, Z. Deep Attentive Anomaly Detection for Microservice Systems with Multimodal Time-Series Data. In Proceedings of the 2022 IEEE International Conference On Web Services (ICWS), Barcelona, Spain, 10–16 July 2022; pp. 373–378.
10. Yang, Z.; Nguyen, P.; Jin, H.; Nahrstedt, K. MIRAS: Model-based Reinforcement Learning for Microservice Resource Allocation over Scientific Workflows. In Proceedings of the 2019 IEEE 39th International Conference On Distributed Computing Systems (ICDCS), Dallas, TX, USA, 7–9 July 2019; pp. 122–132.
11. Al-Doghman, F.; Moustafa, N.; Khalil, I.; Sohrabi, N.; Tari, Z.; Zomaya, A. AI-Enabled Secure Microservices in Edge Computing: Opportunities and Challenges. IEEE Trans. Serv. Comput. 2023, 16, 1485–1504.
12. Erich, F.; Amrit, C.; Daneva, M. A qualitative study of DevOps usage in practice. J. Softw. Evol. Process 2017, 29, e1885.
13. Wang, T.; Zhang, W.; Xu, J.; Gu, Z. Workflow-Aware Automatic Fault Diagnosis for Microservice-Based Applications with Statistics. IEEE Trans. Netw. Serv. Manag. 2020, 17, 2350–2363.
14. Taibi, D.; Lenarduzzi, V.; Pahl, C. Processes, Motivations, and Issues for Migrating to Microservices Architectures: An Empirical Investigation. IEEE Cloud Comput. 2017, 4, 22–32.
15. Calvaresi, D.; Marinoni, M.; Sturm, A.; Schumacher, M.; Buttazzo, G. The challenge of real-time multi-agent systems for enabling IoT and CPS. In Proceedings of the WI'17: International Conference On Web Intelligence, Leipzig, Germany, 23–27 August 2017; pp. 356–364.
16. Bass, L.; Weber, I.; Zhu, L. DevOps: A Software Architect's Perspective; Addison-Wesley Professional: Boston, MA, USA, 2015.
17. Merkel, D. Docker: Lightweight Linux containers for consistent development and deployment. Linux J. 2014, 2014, 2.
18. Bandari, V. A comprehensive review of AI applications in Automated Container Orchestration, Predictive maintenance, security and compliance, resource optimization, and continuous Deployment and Testing. Int. J. Intell. Autom. Comput. 2021, 4, 1–19.
19. Li, J.; Ying, S.; Li, T.; Tian, X. TraceDAE: Trace-Based Anomaly Detection in Micro-Service Systems via Dual Autoencoder. IEEE Trans. Netw. Serv. Manag. 2025, 22, 4884–4897.
20. Xu, S.; Liu, Y. VAE-based Fault Diagnosis for Microservice System. In Proceedings of the 2024 International Conference on Ubiquitous Computing and Communications (IUCC), Chengdu, China, 20–22 December 2024; IEEE: Piscataway, NJ, USA, 2024; pp. 78–85.