International Journal of Advanced Engineering and Technology Research
ISSN: 3106-8677 (Print), ISSN: 3106-8685 (Online) | Vol. 4, No. 2, 2026

Explainable Anomaly Detection in Cloud-Native Systems Through Multimodal Telemetry Fusion

Linnea Sjoberg, Erik Holmstrom *

Department of Computer Science, Lund University, Lund, Sweden

Abstract: Cloud-native services emit operational metrics, event logs, and request traces that describe complementary aspects of failures. This paper presents a lightweight and intrinsically explainable anomaly detector that aggregates these modalities into request windows, converts their deviations from normal behavior into robust scores, and combines them using inverse-dispersion weights. Each alert is accompanied by exact per-modality score contributions, while a separate service-level contrast ranks potential fault locations. The evaluation uses an executed three-process HTTP microservice testbed with five controlled fault mechanisms and two possible fault targets. Forty fault-injection episodes generated 1,440 HTTP requests and 240 non-overlapping telemetry windows. In four repetition-held-out folds, the weighted fusion achieved a pooled AUROC of 0.978 and F1 of 0.896; a trace-only model reached F1 of 0.934. Weighted fusion had the highest AUROC, demonstrating a useful ranking advantage without uniformly dominating thresholded classification. Paired episode bootstrap intervals, modality-pair ablations, and operating-threshold sensitivity quantify these trade-offs. The study also examines fault-family sensitivity and transparent evidence attribution.

Keywords: Anomaly detection, cloud-native systems, microservices, explainability, metrics, logs, distributed tracing, telemetry fusion.

1. Introduction

Cloud-native applications are typically built from independently deployable services whose dependencies spread performance regressions beyond the component where a fault originates [1–3]. Consequently, operational diagnosis requires more than a single scalar alarm. Resource and service metrics describe aggregate health; logs capture discrete runtime events; and distributed traces expose request-path effects [4–6]. Although collecting all three types of telemetry is increasingly practical, deciding how to combine them while providing usable evidence remains an engineering challenge.

Classical outlier detectors and modern deep models address different aspects of abnormal system behavior [7–11], but a complex model can be difficult to audit during incident response. Log-centric approaches characterize event sequences [12–20], while time-series models represent multivariate signals [21–25]. Root-cause studies additionally show that detection and localization are related but distinct tasks [26–37]. In a small monitored service, a method requiring heavy training or a large anomaly corpus can be unnecessary.

We investigate a compact alternative: robust per-modality scoring, a dispersion-calibrated additive fusion rule, and a service-level contrast view. The focus is not to reproduce large-cluster benchmarks but to measure, on actual service executions, what each telemetry source contributes to ranked anomaly detection and explanation.

The contributions are: (i) a low-overhead feature and fusion design with exact additive evidence scores; (ii) a controlled multi-process, multi-fault evaluation with repetition-wise separation of training and testing; and (iii) a comparison of ranking and operating-point performance, supplemented by modality ablations, episode-block bootstrap intervals, threshold sensitivity, fault-family recall, and service attribution.

2. Related Work

2.1. Observability and Multimodal Detection

Distributed tracing emerged as a practical mechanism for understanding cross-service latency [4], and vendor-neutral observability conventions define the complementary roles of traces, metrics, and logs [5]. Log studies have explored invariant mining, parsing, sequence learning and semantic robustness [12–20]. Metric-oriented work ranges from isolation-based anomaly detection [9] to recurrent, variational, and transformer models [21–25]. These methods motivate useful representations, while the present study prioritizes inspectable scores and a bounded feature set over a learned high-dimensional representation.

2.2. Microservice Diagnosis and Explainability

CloudRanger, MicroRCA, Seer, Sage, FIRM, MicroRank, and BARO investigate diagnosis, telemetry-driven resource management, or localization at different levels [30–36]. Recent microservice data resources such as RCAEval, LO2, LO2v2, and AnoMod support broader benchmarking [26–29]. Our controlled testbed is not treated as a replacement for these public datasets. Explanation frameworks including LIME and SHAP emphasize local evidence attribution [38, 39]; interpretability evaluation also requires attention to faithfulness rather than presentation alone [40–42]. Here, attribution follows directly from the scoring equation rather than an auxiliary surrogate.

3. Proposed Method

3.1. Telemetry Windowing and Feature Representation

Three HTTP services are instrumented: a gateway routes each request sequentially through catalog and payment services. Every request produces service latency/status spans,

process-level CPU-time and resident-memory observations, and severity-tagged log events. Six successive requests form one non-overlapping window. For each window w, the representation is x(w) = [xM(w), xL(w), xT(w)] with four scalar features per modality. Window aggregation is intentionally request-based; no claim of uniform wall-clock sampling is made.

The service-level diagnosis view retains each backend's mean latency, CPU time, memory footprint, and alert count. These diagnostics are not inserted into the 12-dimensional detection vector; they are used only for the separate localization analysis.

Table I. Telemetry Features Per Window

Mode	Features used by the detector
Metrics	Request throughput; mean process CPU time; maximum RSS; failed-request fraction
Logs	ERRORs/request; WARNs/request; number of affected services; all alert events/request
Traces	Gateway p95 latency; downstream p95 latency; downstream span spread; error-span fraction

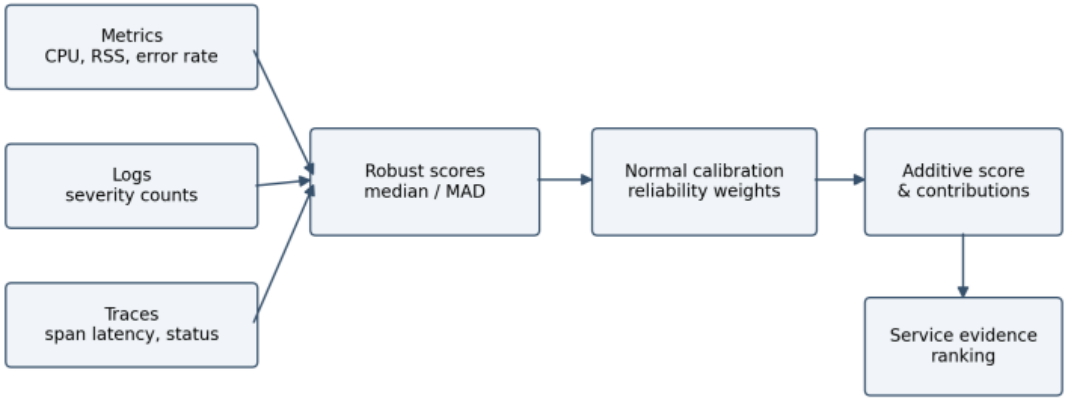


Figure 1. Three telemetry modalities are scored separately, calibrated on normal windows, and fused additively; per-service contrasts provide a second explanatory view.

3.2. Normal-Reference Scoring and Reliability Weights

For feature j, let μj be the median computed from training normal windows. Define a robust feature scale aj as 1.4826 times median absolute deviation when nonzero; otherwise use the positive standard deviation, with a small scale floor when both vanish. The standardized deviation is zj(w) = min{12, |xj(w) − μj|/aj}. Within each modality k ∈ {M, L, T}, the score sk(w) is the arithmetic mean of its two largest feature deviations. This top-two average reduces dilution by unresponsive features while retaining a fixed and comprehensible calculation.

Scores are made comparable by normal calibration: uk(w) = sk(w)/max{Q0.95(sk|normal training), 0.05}. We then assign wk proportional to 1/max{SD(uk|normal training), 0.1}, with Σk wk = 1. The final anomaly score is S(w) = Σk wk uk(w), and an alert occurs when S(w) exceeds the 95th percentile of fused training-normal scores. The procedure requires normal windows only; anomaly labels do not set any threshold or weight.

3.3. Intrinsic Explanations and Service Ranking

Each predicted window is accompanied by the triplet ck(w) = wk uk(w). Since S(w) = cM(w) + cL(w) + cT(w) exactly, the leading contribution identifies which observed modality explains the score under the specified scoring rule. The same equality supports a transparent evidence-removal test: removing a contribution subtracts precisely its value, without approximating a black-box decision.

For retrospective service ranking, the mean post-event increase of each backend's latency, CPU time, and memory is divided by an appropriate pre-event variability scale; the backend's alert count adds discrete evidence. Summing the four nonnegative contrasts yields a service score. This secondary analysis presumes that the pre/post boundary is known for the evaluated episode and ranks only catalog versus payment; it is not a causal identification guarantee.

3.4. Computational Cost and Operational Interface

For K=3 modalities and d=4 features in each modality, producing a window score requires O(Kd) deviations, a constant-time selection of the two largest values in each four-feature group, and three weighted additions. The normal baseline consists of a per-feature median and dispersion estimate plus three score distributions. These summaries are computed only at calibration time. At inference, there is no neural-network optimization, dynamic graph search, or feature-importance sampling. The primary monitoring record contains the current score, the calibrated threshold, three contributions, and the current top-ranked modality. It can be retained alongside raw log and trace identifiers for drill-down.

Separating detector evidence from localization evidence is intentional. A high gateway error rate, for example, can be useful for deciding that an incident exists but misleading as a fault origin because it propagates from downstream calls. The service contrast therefore examines the two backend endpoints directly. When the backend with the larger contrast is reported, the diagnosis is a ranking of observed change relative to that backend's own pre-event behavior. In the experiment the gateway is never injected with a fault, so it is excluded from the candidate set.

All feature deviations are two-sided. This captures both increases (such as request duration) and decreases (such as throughput), but it also means an improvement could receive an anomaly score if sufficiently different from the reference distribution. A production implementation can attach directional semantics to feature families, whereas the present version deliberately retains an uncomplicated, symmetric detector for controlled evaluation.

4. Experimental Design

4.1. Executed Microservice Testbed

The experiments ran locally on a Linux host as three separate Python HTTP-server processes communicating over loopback TCP. The deployment reproduces a small microservice call graph rather than a Kubernetes cluster or production workload. Instrumentation collected actual CPU-time differences, resident memory measurements, severity-tagged process logs, and measured request/spanning latency. Faults were introduced into either catalog or payment while gateway retained a fixed routing topology.

Five fault mechanisms were tested: an approximately 14-ms extra service delay; deterministic HTTP 503 responses on selected calls; a CPU busy loop of approximately 7 ms; retained 420-kB memory buffers on successive calls; and a packet-loss proxy represented by delayed HTTP 504 responses. The last mechanism models a failed request at the application level and does not manipulate physical network packets. Each episode contains three normal windows followed by three injected windows. The injected-regime label is applied to every post-injection window even if only some of its constituent requests are affected.

Table II. Measured Experiment Scope

Item	Value
Service processes / backend targets	3 / 2
Fault types / repetitions	5 / 4
Fault episodes / HTTP requests	40 / 1,440
Windows / windows per episode	240 / 6
Requests per window	6
Held-out testing	4 repetition-based folds
Training normals in each fold	90 windows
Test windows in each fold	60 (30 normal, 30 injected)

4.2. Comparison Protocol

All models are evaluated with leave-one-repetition-out validation. In each fold, the normal windows from the other three repetitions are used for training and score-threshold calibration; complete episodes in the held-out repetition provide testing data. This yields disjoint training and test episodes and the same 120 normal and 120 injected test windows across methods. The compared configurations are metric-only, log-only, trace-only, equal-weight late fusion, the proposed inverse-dispersion late fusion, and an early-fusion Isolation Forest. The latter uses 120 trees, at most 128 normal training examples per tree, and fixed random seed 17 [9, 45].

We report AUROC, area under the precision–recall curve (AUPRC), precision, recall, F1 and false-positive rate [43], [44]. Reported pooled AUROC/AUPRC combine held-out scores across folds; fold-mean AUROC is separately examined because score calibration can differ between folds. All figures and numbers below are calculated from saved request/window tables and prediction records.

4.3. Telemetry Collection and Analysis Controls

Service handlers write compact JSON event logs with episode identifier, request identifier, timestamp, severity, and event type. The gateway obtains downstream JSON span information over actual HTTP calls and combines it with its own measured span. The experiment runner records both per-request measurements and window summaries. These records are not generated by drawing numerical observations from a distribution; the intervention schedule is deterministic, while performance observations originate from executed handlers and the local operating system. To make reruns possible, the provided Python program constructs the three services, exercises every condition, aggregates all records, trains every detector, and exports the scores and figures [46-51].

The comparison reuses a single fixed observation table for every method, eliminating differences caused by separate load runs. Training uses only windows preceding injection from episodes outside the held-out repetition. Test episodes preserve their local normal-to-fault chronology, and no post-injection observation is used to determine normalization constants or operating thresholds. The paper provides aggregate statistics and the exact fold protocol rather than listing every measured request or every handler event in the main text [52-57].

The fault controls serve specific measurement purposes. Added latency tests whether traces reveal delayed downstream service spans; CPU activity tests process-time metrics; retained buffers exercise memory-sensitive signals; and 503/504 responses test the overlap among status metrics, warning/error logs, and failed spans. Because different fault families illuminate different telemetry sources, combining them is informative even when one modality dominates the aggregate F1 metric.

5. Results and Analysis

5.1. Detection Performance

Table III. Pooled Held-Out Detection Results

Method	AUROC	AUPRC	F1	FPR
Metrics	0.878	0.841	0.709	0.108
Logs	0.700	0.700	0.571	0.000
Traces	0.958	0.962	0.934	0.075
Early IF	0.934	0.943	0.802	0.033
Equal fusion	0.977	0.977	0.896	0.150
Weighted fusion	0.978	0.979	0.896	0.150

The weighted late-fusion method reached pooled AUROC 0.978 and AUPRC 0.979, compared with 0.958 and 0.962 for traces alone. Its F1 was 0.896 (112 true positives, 18 false positives, eight false negatives), whereas the trace-only configuration reached the higher F1 of 0.934 (113 true positives, nine false positives, seven false negatives). Equal fusion produced very similar classification results (AUROC 0.977, F1 0.896). Thus, fusion improved ranking of normal versus injected windows in this setting, but additional modalities increased false alerts at the selected unsupervised threshold.

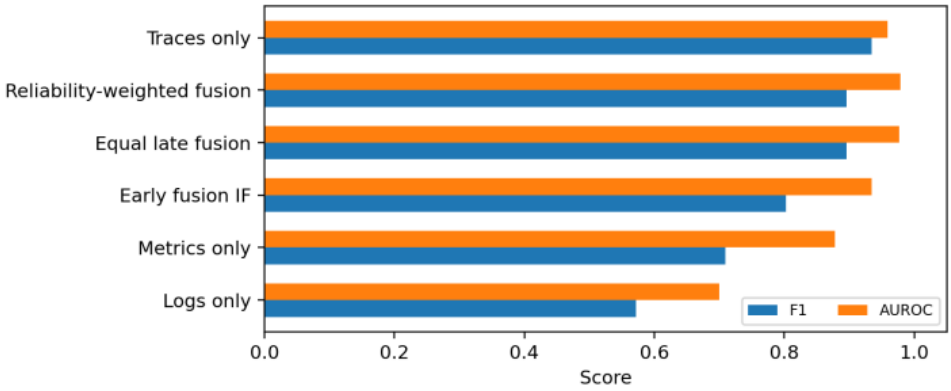


Figure 2. Pooled held-out AUROC and F1 by detection configuration.

Across the four folds, mean AUROC was 0.987 ± 0.015 for weighted fusion and 0.969 ± 0.015 for traces alone (mean ± sample standard deviation). In contrast, log-only features detected only 48 of 120 injected windows, consistent with the latency, CPU, and memory faults often producing normal-severity logs. Isolation Forest yielded F1 0.802, showing that concatenation into a single unsupervised model was not preferable to the compact scoring rule for these data.

5.2. Modality Ablation and Paired Uncertainty

Table IV. Modality-Pair Ablation (Pooled Test Windows)

Fusion configuration	AUROC	AUPRC	F1	FPR
M+L	0.910	0.916	0.796	0.108
M+T	0.973	0.971	0.896	0.150
L+T	0.964	0.972	0.934	0.075
Equal 3-way	0.977	0.977	0.896	0.150
Weighted 3-way	0.978	0.979	0.896	0.150

The two-source comparisons isolate the contribution of each modality. Metrics plus logs reached AUROC 0.910; adding traces to either source increased it to 0.973 (M+T) or 0.964 (L+T). Logs plus traces retained the same F1 as traces alone (0.934). Including metrics in the three-way fusion improved AUROC to 0.978 but lowered F1 to 0.896 at the default threshold. This pattern identifies rank quality versus false-alert cost as the central model-selection trade-off.

To account for windows sharing an injected episode, we resampled all six windows together within each of the ten fault-by-target strata (four episodes per stratum; 5,000 paired bootstrap replicates, fixed seed). Relative to trace-only scoring, weighted fusion changed pooled AUROC by +0.0197 (95% percentile interval [+0.0007, +0.0388]) and F1 by -0.0379 (interval [-0.0745, -0.0018]). These intervals describe sampling variability across the observed controlled episodes, rather than variability among independent production environments.

5.3. Threshold Sensitivity and Fold Comparison

The default operating point is the 95th percentile of training-normal scores, determined without fault labels. The ranking advantage of fusion does not guarantee an F1 advantage because each modality changes the calibration distribution. To expose this effect, we also evaluated normal-score quantiles 0.90, 0.975, and 0.99 using the same held-out test observations; the alternative quantiles were not selected using anomaly labels.

Table V. Weighted Fusion: Training-Normal Quantile

Quantile q	Precision	F1	FPR
0.900	0.858	0.906	0.158
0.950	0.862	0.896	0.150
0.975	0.915	0.908	0.083
0.990	0.945	0.900	0.050

For example, the 0.975 quantile reduced the held-out false-positive rate from 0.150 to 0.083 (18 to 10 false alarms) while yielding F1 0.908. The 0.99 quantile gave FPR 0.050 and F1 0.900. These values emphasize that the nominal 95th training percentile is not a guarantee of 5% test false alarms under repetition-to-repetition variation.

Pooling the four held-out prediction tables yields the AUROC values in Table III. The corresponding fold-mean AUROC values are 0.987 for weighted fusion, 0.987 for equal fusion, 0.969 for traces, 0.936 for early Isolation Forest, 0.938 for metrics, and 0.700 for logs. Per-fold ranking removes between-fold score-calibration effects that can influence pooled AUROC; both summaries are retained to make score aggregation explicit.

5.4. Fault-Family Sensitivity

Table VI. Weighted Fusion Recall by Injection Type

Fault mechanism	Injected windows	Recall
latency	24	1.000
http error	24	1.000
cpu hog	24	1.000
memory leak	24	0.667
packet loss	24	1.000

Weighted fusion detected every injected window of the delay, HTTP-error, CPU-busy, and application-level packet-loss-proxy conditions; memory retention was harder (16/24 detected windows, recall 0.667). The latter case did not necessarily induce a large short-window latency or error change, explaining the weaker sensitivity. Fault exposure was the ground-truth label, not a manually reviewed operational incident.



Figure 3. Detection recall by intervention type for the weighted late-fusion model.

5.5. Inspectable Evidence and Service Contrast

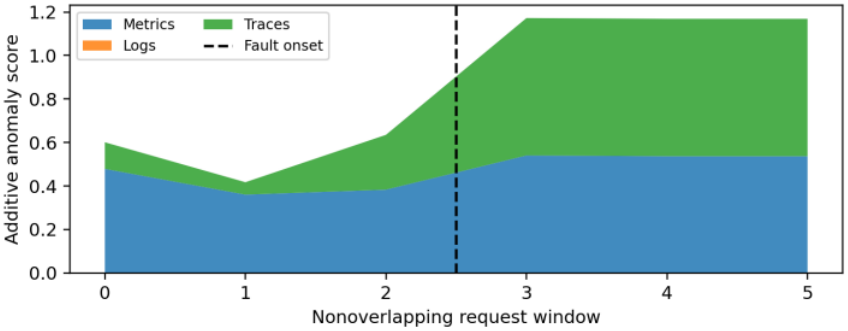


Figure 4. Actual modality contributions across six windows of the first latency-injection episode; dashed line marks injection onset.

Fig. 4 plots an actual catalog-delay episode. Every plotted ordinate is a sum of three stored contributions; the nonzero trace component grows with measured downstream latency. The explanation is therefore a numerical decomposition of the alert score rather than a textual post-hoc rationale. Across all 40 held-out episodes, the retrospective two-candidate service contrast ranked the injected backend first in 40/40 cases. Weighted fusion first flagged an injected window at the onset in 37/40 episodes, one window later in one episode, and two windows later in two episodes; these are request-window steps, not wall-clock seconds. The 40/40 service contrast result remains restricted to two known candidates and a known episode boundary, not causal root-cause inference in arbitrary dependency graphs.

6. Discussion and Validity

The results support two practical observations. First, traces supplied the strongest single-modality thresholded classification, while weighted three-source fusion improved ranking but was sensitive to the chosen unsupervised threshold. Second, the additive construction makes evidence inspection inexpensive: the stored contribution vector is itself sufficient to recover the score, identify its dominant source, and perform exact modality-removal accounting. This is narrower than causal explanation, but directly usable for triaging whether a window is principally characterized by resource, log, or latency evidence [38–42].

There are clear boundaries to interpretation. The testbed is a three-service local topology, not an orchestrated production cluster. Controlled interventions use deterministic schedules, sequential request generation, and compact feature windows; mixed simultaneous failures, noisy tenants, evolving traffic, and longer dependency chains were not tested. All post-onset windows are labeled as exposed to the injection, and the network-failure condition is a service-level HTTP proxy. The 100% service-ranking result benefits from only two candidates and a known pre/post split. The present experiment also does not establish superiority over specialized neural architectures or industrial root-cause tools on their respective public benchmarks [21–37]. These differences delimit the scenarios to which the measured values apply.

An operational extension can implement the same score in a streaming collector, recalibrate robust normal statistics by workload regime, and attach actual trace identifiers and log-event samples to each contribution. Because feature extraction and fusion are simple, this can be studied independently of model retraining and without an extensive labeled anomaly corpus.

7. Deployment Implications

A lightweight score-based method fits a small observability installation because the system does not need an incident-label pipeline or a high-capacity accelerator. At configuration time, operators must still select a reference workload regime, define the request window, and review whether the chosen metrics carry meaningful variation. The reliability rule reduces the influence of modalities with unstable normal scores but cannot by itself correct persistent bias, missing spans, or delayed log delivery. Consequently, per-modality data availability and timestamp alignment should be recorded when adapting the method to an asynchronous collector.

The study also illustrates why interpretability should be expressed at more than one resolution. A modality contribution explains how the anomaly score was obtained; a service contrast proposes where the most conspicuous change occurred; and the original log and span records show concrete supporting events. These are successive levels of evidence, not interchangeable explanations. A backend with a high contrast is a reasonable inspection target, yet a dependency-driven propagation pattern or shared infrastructure fault could produce the same observation. Causal localization would require additional assumptions or interventions [30]–[37].

The experimental observation that trace-only classification

has the highest F1 argues for retaining simple single-modality baselines even when introducing multimodal models. If alert costs favor precision under an operating threshold, the trace baseline is a defensible default for this particular testbed. If ranking is central—for example, when analysts inspect a small number of highest-scoring windows—the fused score is a reasonable alternative. The selection should follow the operational objective rather than an assumption that every extra data source must improve every metric.

8. Conclusion

We evaluated a compact explainable anomaly detector using 1,440 executed HTTP requests from a three-service composition. Inverse-dispersion fusion achieved pooled AUROC 0.978, compared with 0.958 for trace-only detection; traces alone and logs-plus-traces both attained the highest F1 (0.934). Modality ablations, episode-block bootstrap intervals, and threshold sensitivity clarify the trade-off between ranking performance and false alarms. Exact score decomposition and retrospective service contrasts provide directly inspectable telemetry evidence.

References

[1] Liang, Y., Jiao, Y., Ping, W., Fan, H., & Han, X. (2026). Adaptive event-driven labeling: A neuro-symbolic multiagent framework for causal inference in non-stationary time series. IEEE Access.

[2] Qin, Y., Rhee, M., Teng, D., Zhang, C., & Zi, B. (2026). Neuro-symbolic constraint verification for LLM-driven internet finance transaction execution. Scientific Reports.

[3] Jiao, Y., Fan, H., Yue, X., Ping, W., Sun, T., & Wang, J. (2026). Dynamic heterogeneous graph contrastive learning for uncovering collusive financial fraud. Scientific Reports.

[4] Shen, A., & Liu, W. (2026). Budget-Aware Adaptive Test-Time Compute with Hierarchical Self-Verification for Semantic ID Generative Recommendation. Computers.

[5] Cao, J., Zheng, W., Ge, Y., & Wang, J. (2025). DriftShield: Autonomous fraud detection via actor-critic reinforcement learning with dynamic feature reweighting. IEEE Open Journal of the Computer Society, 6, 1166–1177.

[6] Mai, N. T., Cao, W., & Fang, Q. (2025). A study on how LLMs (eg GPT-4, chatbots) are being integrated to support tutoring, essay feedback and content generation. Journal of Computing and Electronic Information Management, 18(3), 43–52.

[7] Zhang, C., & Wang, Y. (2024). A Data-Driven Multi-Objective Production Scheduling Framework for Battery Pack Manufacturing Under Material Arrival Uncertainty. AI and Data Science Journal, 1(1), 86–93.

[8] Lin, Y., Yue, X., Bian, M., Zhu, R., & Teng, D. (2026). Subgraph-level temporal graph transformer with laundering-motif prototypes for anti-money laundering on blockchains. IEEE Access.

[9] Li, P., Feng, E., Wang, G., & Mo, T. (2024). Hybrid Deep Reinforcement Learning and Integer Optimization for Resource Scheduling In Heterogeneous Ai Computing Clusters. Innovation and Technology Studies, 1(1), 48–56.

[10] Yue, X., & Zhu, R. (2024). An explainable hybrid machine learning framework for cash-flow-conditioned multi-horizon liquidity risk early warning: An SME-oriented benchmark study. World Journal of Management Science, 2(1), 63–72.

[11] Wang, J., Tan, Y., Jiang, B., Wu, B., & Liu, W. (2025). Dynamic marketing uplift modeling: A symmetry-preserving framework integrating causal forests with deep reinforcement learning for personalized intervention strategies. Symmetry, 17(4), 610.

[12] Chen, J., Liu, J., Liang, Y., & Zhou, M. (2026). HeteroGCL: A Heterogeneous Graph Contrastive Learning Framework for Scalable and Sustainable Cryptocurrency AML. Applied Sciences, 16(6), 2860.

[13] Teng, D. (2025). Triad: Deadline- And Reliability-Aware Scheduling for Post-Earthquake Structural Assessment Workflows. World Journal of Engineering Research, 3(6), 55–66.

[14] Shang, W., & Chen, T. (2024). PrefixKD-Edge: Real-time locomotion-intent prediction with prefix self-distillation on lightweight temporal networks. Innovation and Technology Studies, 1(1), 30–38.

[15] Teng, D. (2025). Rsrbench: Reproducibility and Performance Benchmarking for Regional Seismic Risk Workflows. Academic Journal of Architecture and Civil Engineering, 3(1), 40–52.

[16] Rhee, M., Zou, J., & Qin, Y. (2024). CLARITY: A closed loop framework for duplicate aware failure prediction and budgeted automated remediation in cloud infrastructure. AI and Data Science Journal, 1(1), 76–85.

[17] Fan, H., Liang, Y., Jiao, Y., Ping, W., & Lin, Y. (2026). An Explainable Temporal Machine Learning Framework for Financial Fraud Risk Prediction in Listed Institutions. Highlights in Business, Economics and Management, 70, 173–185.

[18] Shen, A., & Huang, S. (2022). A Systems-Aware Framework for Intelligent Large-Scale Recommendation Models. Journal of Computer Science and Electrical Engineering, 4(1), 36–44.

[19] Zhu, R., & Yue, X. (2024). An explainable machine learning framework for predicting dividend sustainability and financial risk of US REITs under interest-rate shocks. Journal of Trends in Finance and Economics, 1(2), 48–57.

[20] Zhao, X., Liu, J., Wang, Y., & Wang, J. (2026). CryptoMamba-SSM: Linear complexity state space models for cryptocurrency volatility prediction. IEEE Open Journal of the Computer Society, 7, 226–243.

[21] Mo, T., Zhang, C., Zou, J., Guo, Z., & Rhee, M. (2026). Self-Evolving AI Agents with Dual Memory for Automated Software Testing and Bug Localization. IEEE Access.

[22] Qin, Y., Zou, J., & Rhee, M. (2022). An AI-driven intelligent transaction execution framework for internet finance platforms: Cost-sensitive real-time integrity interdiction. Eurasia Journal of Science and Technology, 4(2), 36–44.

[23] Chen, J., Cui, Y., Zhang, X., Yang, J., & Zhou, M. (2024). Temporal convolutional network for carbon tax projection: A data-driven approach. Applied Sciences, 14(20), 9213.

[24] Fang, Q., & Liu, W. (2025). HARLA-ED: Resolving Information Asymmetry and Enhancing Algorithmic Symmetry in Intelligent Educational Assessment via Hybrid Reinforcement Learning. Symmetry, 18(1), 58.

[25] Zhang, F., Shen, A., & Huang, S. (2026). TRIAD: Co-designing mixed-precision quantization, operator fusion, and on-chip memory reuse for real-time vision–language–action inference on automotive NPUs. Results in Engineering, 113673.

[26] Guo, Z., Chen, T., & Shang, W. (2022). Class-conditional intermediate-domain adversarial adaptation for cross-device acoustic scene classification. Journal of Computer Science and Electrical Engineering, 4(1), 26–35.

[27] Zhou, F. (2026). Moment-Constrained Side-Force Allocation for Wind-Assisted Merchant Ships: Hydrodynamic Performance of Retractable Anti-Leeway Fins and Flap Rudders. Innovation and Technology Studies, 3(1), 27–41.

[28] Wang, G., Li, P., Feng, E., Shen, A., & Rhee, M. (2026). A dual-process memory-augmented large language model agent for explainable sequential recommendation. IEEE Access.

[29] Zhang, F., & Yang, J. (2022). Accuracy-Aware Model Compression for Real-Time Visual Perception Under Resource Constraints. Journal of Computer Science and Electrical Engineering, 4(1), 45–52.

[30] Bian, M., Wang, Y., & Lin, Y. (2024). Explainable Supplier Risk Prediction for Critical Telecommunications Infrastructure Procurement: An Empirical Xgboost–Shap Framework. Social Science and Management, 1(1), 64–77.

[31] Fang, Q. (2024). Bias-Aware Large Language Models for Equitable Educational Feedback and Assessment. Trends in Social Sciences and Humanities Research, 2(1), 41–50.

[32] Zhang, F., & Yang, J. (2024). Robustness-Aware Model Compression for Multimodal Perception Under Challenging Conditions. AI and Data Science Journal, 1(1), 103–111.

[33] Feng, E., Li, P., Mo, T., & Wang, G. (2022). Prism: Prediction-Guided Risk-Aware Deep Reinforcement Learning for Energy-Aware Dynamic Resource Scheduling in Distributed Cloud Data Centers. Journal of Computer Science and Electrical Engineering, 4(1), 53–61.

[34] Sun, T., Yang, J., Li, J., Chen, J., Liu, M., Fan, L., & Wang, X. (2024). Enhancing auto insurance risk evaluation with transformer and SHAP. IEEE Access, 12, 116546–116557.

[35] Guo, Z., Fang, Q., Yang, J. S., & Rhee, M. (2026). Closed-loop audio neuroregulation for ADHD: Graph-based attention decoding and rhythm entrainment. IEEE Access.

[36] Yue, X., Yang, J., & Liu, W. (2026). FraudDebate-Agent: A multi-agent LLM framework with an evidence-based debate mechanism for financial statement fraud detection. Mathematics, 14(15), 2695.

[37] Mo, T., Wang, G., Feng, E., & Li, P. (2023). Execution-Feedback-Guided Context Selection for Large Language Model-Based Software Testing and Debugging. World Journal of Information and Knowledge Management, 1(1), 55–64.

[38] Chen, T., Shang, W., & Guo, Z. (2024). Xsla: An SLA-aware heterogeneous XPU scheduling framework for edge AI inference. World Journal of Information Technology, 2(2), 48–59.

[39] Fang, Q. (2024). True-Ews: Trustworthy Early Warning with Equalized Conformal Guarantees for Equitable Student Support In U.s. Education. Educational Research and Human Development, 1(2), 44–54.

[40] Zhang, H. (2024). Graph-Aware Multi-Task Learning for Early Prediction of Timing Violations and Routing Congestion In Digital Ic Physical Design. World Journal of Information Technology, 2(1), 13–20.

[41] Shen, A. (2023). Compute-Efficient Scaling of Large Sequential Recommendation Models Under Dynamic User Behavior. World Journal of Information Technology, 1(1), 62–71.

[42] Wang, G., Li, P., Feng, E., & Mo, T. (2024). Uncertainty-Aware Llm-As-A-Judge Evaluation for Reliable Search and Recommendation Systems: Calibrated Confidence And Impact-Aware Selective Human Review. AI and Data Science Journal, 1(1), 112–121.

[43] Chen, J., Han, X., Zhang, X., & Yang, Y. (2023). Exact Concept-Level Explanations for Early Detection of Financial Misstatements. Eurasia Journal of Science and Technology, 5(2), 34–48.

[44] Liang, Y., & Jiao, Y. (2024). Real-time financial fraud detection using a feature-light temporal dynamic graph neural network with contrastive learning. Journal of Trends in Finance and Economics, 1(1), 76–84.

[45] Jiao, Y., & Liang, Y. (2024). An Integrated Process-Mining And Explainable Machine-Learning Framework for Bottleneck And Internal-Control Risk Detection in the Multi-Entity Financial Close. Social Science and Management, 1(3), 96–107.

[46] Lin, Y., Wang, Y., & Bian, M. (2024). Xor-Ew: An Explainable and Cost-Aware Artificial Intelligence Framework for Operational Risk Early Warning In Financial Institutions. AI and Data Science Journal, 1(1), 94–102.

[47] Wang, Y., Bian, M., & Lin, Y. (2022). An Explainable Temporal Graph Neural Network for Disruption Prediction and Risk Propagation in Multi-Tier Supply Chains. Journal of Computer Science and Electrical Engineering, 4(1), 17–25.

[48] Wang, J., Liu, J., Zheng, W., & Ge, Y. (2025). Temporal heterogeneous graph contrastive learning for fraud detection in credit card transactions. IEEE Access.

[49] Wang, J., Ge, Y., Liu, J., & Jiang, B. (2024). Early Warning Of Liquidity Stress In U.s. Banks With Seasonally Aligned Frailty Hazards and Delayed-Feedback Conformal Alarms. Journal of Trends in Finance and Economics, 1(1), 85–100.

[50] Ping, W., & Fan, H. (2024). Disclosure Quality and Valuation Risk in Level 3 Investments: A Disclosure-Implied Risk Framework for U.s. Business Development Companies. Social Science and Management, 1(1), 78–87.

[51] Mai, N. T., Fang, Q., & Cao, W. (2025). Measuring student trust and over-reliance on AI tutors: Implications for STEM learning outcomes. International Journal of Social Sciences and English Literature, 9(12), 11–17.

[52] Chen, J., Liang, Y., Liu, J., & Zhou, M. (2026). Temporal Transformer with Conditional Tabular GAN for Credit Card Fraud Detection: A Sequential Deep Learning Approach. Mathematics, 14(7), 1183.

[53] Wang, J., Liu, J., Ge, Y., & Jiang, B. (2023). Risk-Aware Ensemble Learning for Credit Default Prediction Under Economic Uncertainty. World Journal of Economics and Business Research, 1(1), 18–32.

[54] Ping, W., & Fan, H. (2023). Beyond Isolated Ratios: Accrual–Revenue–Cash Flow Indicators and Interaction-Aware Learning for Detecting Financial Reporting Irregularities In U.s. Public Companies. World Journal of Economics and Business Research, 1(1), 46–55.

[55] Fan, H., Jiao, Y., Wang, M., Chen, J., & Yang, S. (2026). Fraud learns too: Continual graph learning under strategic adversarial drift in dynamic networks. Scientific Reports.

[56] Zou, J., Qin, Y., & Rhee, M. (2024). Beyond model scale: A task-aware reliability benchmark for large language models in clinical decision support. Innovation and Technology Studies, 1(1), 39–47.

[57] Teng, D., Rhee, M., Qin, Y., Zi, B., & Liu, W. (2026). SW-SpeedDLM: Sliding Window Speculative Decoding for Diffusion Language Models Under Long Context Constraints. Mathematics, 14(12), 2137.