CoLMAD: Cost-Efficient LLM-Assisted Time Series Anomaly Detection for Industrial Monitoring

Shenglin Zhang*, Xiaoyu Feng*, Wenwei Gu*, Yongqian Sun*, Yu Kang†, Hailin Zhang*, Jiacheng Zhang*
Yong Xu†, Yuxin Wu*, Dan Pei‡, Lauren Zhang§, Cong Chen§, Yingnong Dang§
*Nankai University, China †Microsoft Research, China ‡Tsinghua University, China §Microsoft Research, United States

***Abstract*—Time series anomaly detection (TSAD) is a core reliability function in production monitoring systems. In practice, lightweight detectors are scalable but often produce false alarms and provide limited explanations, while directly applying large language models (LLMs) to every monitoring window is too slow and costly for online deployment. We present CoLMAD, a cost-efficient LLM-assisted TSAD framework for industrial monitoring. CoLMAD uses lightweight detectors for continuous candidate generation and selectively invokes an LLM validator to refine suspicious windows. The validator compresses numerical sequences into structure-preserving segments, translates them into semantic descriptions, and applies anomaly-aware prompting to produce both decisions and explanations. We evaluate CoLMAD on three public datasets and a large-scale production monitoring dataset. CoLMAD achieves F1 scores of 0.979, 0.987, 0.989, and 0.955, while invoking the LLM for only 0.5% of cases and reducing estimated daily cost from 87.92 USD to 0.08 USD compared with a direct LLM-based baseline.**

***Index Terms*—Time Series Anomaly Detection, Large Language Models, Industrial Monitoring, Interpretability**

I. Introduction

Time series anomaly detection (TSAD) is a core reliability function in production monitoring systems. Modern cloud services continuously emit high-frequency metrics such as latency, throughput, error rate, CPU utilization, and memory usage. These metrics are often the first observable signals of service degradation, so TSAD directly affects incident response: a missed anomaly can delay mitigation, whereas excessive false alarms can quickly overwhelm on-call engineers and reduce trust in the monitoring system [1, 2, 3, 4].

Despite extensive research on TSAD, industrial deployment remains difficult. Statistical and rule-based detectors are still widely used because they are lightweight, easy to maintain, and fast enough for online alerting [5, 6, 7, 8, 9]. However, they often rely on fixed assumptions and manually tuned thresholds, making them brittle under concept drift, noisy signals, local spikes, and long-term distribution shifts [10, 11, 12]. Deep learning and foundation-model approaches improve representation ability, but they may require training data, tuning, and specialized serving resources, and their outputs remain difficult for engineers to verify [13, 14, 15, 16, 17, 18, 19, 20, 21].

Large language models (LLMs) offer a new opportunity for TSAD because they can follow domain instructions, combine evidence, and generate human-readable explanations [22, 23, 24, 25, 26, 27]. However, directly using LLMs as always-on detectors is not practical in production monitoring. Raw numerical sequences are token-inefficient, generic LLMs do not naturally contain TSAD-specific priors, and per-window LLM inference introduces unacceptable latency and cost. Table I summarizes the resulting gap: existing paradigms cover only part of the practical TSAD requirement space.

TABLE I
Coverage of practical TSAD requirements by method category.

Method	Adapt.	Cold-start	Scale	Interp.
Statistical methods	✗	✓	✓	✗
Supervised learning	✗	✗	✗	✗
Unsupervised learning	✗	✓	✗	✗
Foundation models	●	✓	●	✗
LLMs	●	✓	✗	✓
CoLMAD	✓	✓	✓	✓

✓: covered, ✗: not covered, ●: partially or potentially covered.

This paper presents CoLMAD, a cost-efficient LLM-assisted TSAD framework designed for industrial monitoring. Instead of replacing lightweight detectors with LLMs, CoLMAD uses lightweight detectors as always-on candidate generators and invokes an LLM only as a selective validator for suspicious windows. The validator compresses candidate windows into structure-preserving segments, converts them into semantic descriptions, and applies anomaly-aware prompting to refine decisions and produce structured explanations.

We evaluate CoLMAD on three public datasets and one production monitoring dataset collected from a large-scale service environment. CoLMAD achieves F1 scores of 0.979, 0.987, 0.989, and 0.955 on the four datasets. On the production dataset, it obtains 0.913 precision and 1.000 recall. More importantly for deployment, only 0.5% of windows require LLM validation, reducing estimated daily cost from 87.92 USD to 0.08 USD compared with a direct LLM-based baseline. Our contributions are:

- We identify the industrial challenges that prevent direct LLM-based TSAD from being deployable, including structural mismatch, missing TSAD priors, weak sensitivity to local perturbations, and inference cost.
- We propose a hybrid architecture that combines always-on lightweight candidate generation with selective, structure-aware LLM validation.
- We provide empirical evidence on public and production monitoring data, including complete precision/recall/F1 comparisons, deployment-oriented cost analysis, and case studies that show how structured explanations support engi-

neer verification.

II. Industrial Challenges and Design Requirements

CoLMAD targets univariate metric streams collected from production services, such as resource utilization, latency, throughput, and service-level indicators. These streams are monitored continuously with sliding windows. Each decision must be fast enough for online alerting, and each alert should include evidence that an engineer can inspect. In this setting, simply asking an LLM to read every raw window is attractive for interpretability, but it is not a deployable design. Fig. 1 summarizes four challenges that motivate CoLMAD.

C1: Numerical structure mismatch and token inefficiency. LLMs process inputs as discrete tokens, whereas time series contain continuous numerical structures. Directly tokenizing raw values produces long prompts and can break local continuity, trend shape, and interval boundaries. For industrial metrics, this is especially harmful because small structural changes can correspond to serious service degradation.

C2: Lack of TSAD-specific detection priors. Generic language knowledge does not define what is normal for a monitored metric. TSAD decisions require knowledge about normal fluctuation ranges, periodic behavior, local spikes, long abnormal plateaus, and common detector failure modes. Without explicit priors, an LLM may provide plausible but unreliable judgments.

C3: Insensitivity to short-term perturbations. LLMs are strong at global context reasoning, but local anomalies can be short and subtle. A spike that is highly salient relative to its local neighborhood may appear less important when embedded in a long sequence. Industrial monitoring cannot ignore such local events because short bursts may indicate overload, packet loss, or transient service failures.

C4: Scalability and operational cost bottlenecks. Production systems generate large numbers of windows. Per-window LLM validation introduces high latency, API cost, and dependence on a remote model service. This conflicts with the reliability requirement that the alerting path remain lightweight and predictable.

These challenges lead to four design requirements. First, the system should preserve high recall while reducing false alarms, because missed anomalies and alert fatigue both carry operational cost. Second, it should support cold-start and non-stationary metrics without requiring extensive labeled data. Third, it should avoid per-window LLM invocation and keep the always-on path lightweight. Fourth, it should produce actionable explanations, including the normal pattern, anomalous interval, detector reliability, and reason for the final decision. CoLMAD follows these requirements by using lightweight detectors for scalable candidate generation and reserving LLM reasoning for sparse, high-value validation cases.

III. CoLMAD Design

CoLMAD is a two-stage industrial workflow (Fig. 2). The first stage keeps the monitoring path scalable by applying lightweight detectors to every window. The second stage invokes an LLM validator only for suspicious candidates. This design makes LLMs a controlled validation and explanation component, rather than a full replacement for scalable TSAD detectors.

Lightweight candidate generation. The candidate generator uses low-cost statistical or machine-learning detectors to scan every sliding window. In our deployment-oriented setting, this stage is configured to favor recall: it should surface suspicious intervals even if some are false positives. CoLMAD therefore treats detector outputs as weak evidence, not final decisions.

Structure-preserving compression. Direct raw-value prompting is both expensive and noisy. CoLMAD compresses each candidate window into key segments that preserve turning points, abrupt changes, and trend transitions. Unlike generic down-sampling, the goal is not only to shorten the sequence, but also to retain anomaly semantics such as local spikes, sustained shifts, and sharp recovery boundaries. This reduces token usage while keeping the evidence needed for validation.

Language-guided transformation. The compressed segments are converted into structured textual descriptions. Each segment records the index range, value range, direction, and magnitude of change. This step bridges numerical signals and language-based reasoning: the LLM receives an auditable description of the metric behavior rather than a long list of raw values.

Anomaly-aware validation and reporting. The LLM prompt combines the transformed window, the lightweight detector result, TSAD-specific normal/anomaly definitions, and known detector error mechanisms. The LLM is asked to judge whether the detector result is reliable and, if needed, re-evaluate the candidate window. The final output is a structured report containing the normal pattern, final decision, anomalous interval, and reason. These fields are intentionally aligned with what engineers need during alert verification.

IV. Evaluation

A. Experimental Setup

Datasets. We evaluate CoLMAD on four datasets (Table III). D1 is the A2 benchmark from Yahoo [28]. D2 and D3 are the Tweets and AWS subsets from NAB [29]. D4 is our production monitoring dataset collected from a large-scale service environment over one month and annotated by experts. We use the first 30% of each dataset for training learning-based baselines and the remaining 70% for evaluation. Methods without training are evaluated directly on the test portion. The sliding window size is 400.

Metrics and baselines. We report precision (Pre), recall (Rec), and F1. Following common TSAD practice, we sweep thresholds and report the best F1 with point adjustment for contiguous anomaly segments. We compare against statistical methods, semi-supervised and deep TSAD models, time-series foundation models, and LLM-based baselines: LOF [8],

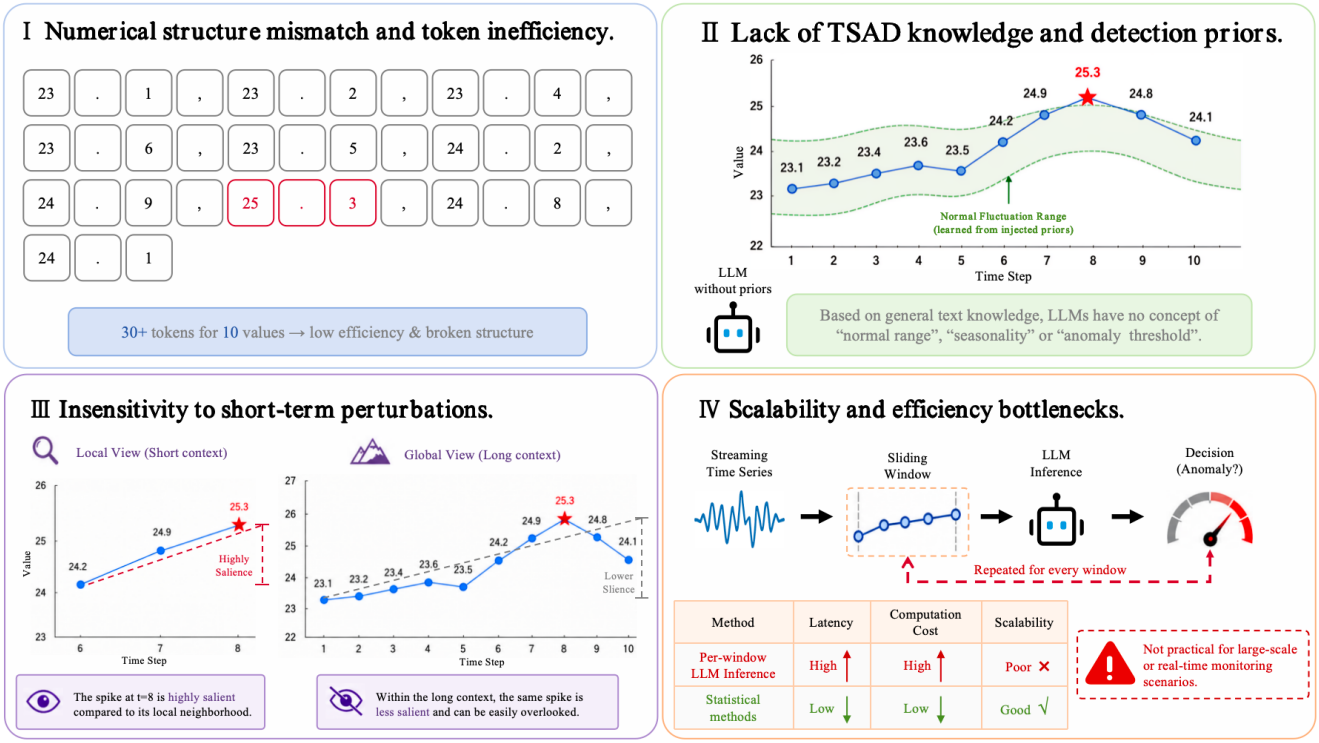

Fig. 1. Practical challenges of directly applying LLMs to industrial TSAD. The four panels illustrate token-level structural mismatch, missing TSAD priors, weak sensitivity to local perturbations, and the cost/latency bottleneck of per-window LLM inference.

iForest [9], OCSVM [30], PUAD [4], Bagel [13], AnoTransfer [31], Donut [14], TimesNet [18], Informer [32], Timer [19], Moirai [21], GPT4TS [22], and LLMAD [25].

B. Detection Effectiveness

Table II reports the complete detection results across D1-D4. CoLMAD achieves the best F1 on all datasets: 0.979 on D1, 0.987 on D2, 0.989 on D3, and 0.955 on D4. The production dataset D4 is the most important for industrial applicability because it contains noisier and more non-stationary metric streams. On D4, CoLMAD obtains 0.913 precision and 1.000 recall, outperforming the strongest baseline F1 of 0.845. This indicates that selective LLM validation improves precision while preserving the high recall required by monitoring scenarios.

From an operational perspective, the D4 result is more important than the public-dataset average. In production monitoring, precision and recall correspond to two different costs: false positives consume engineer attention and gradually reduce trust in the alerting system, while false negatives may delay incident mitigation. CoLMAD improves precision over lightweight detectors while keeping recall at 1.000 on D4, which suggests that the LLM validator mainly acts as a false-alarm filter rather than a lossy second-stage gate. This behavior matches our deployment goal: the first stage should cast a wide net, and the second stage should refine suspicious windows into fewer, better explained alerts.

C. Deployment Cost, Latency, and Explanations

A direct LLM-based detector is impractical for online monitoring because every window would incur token cost and API latency. CoLMAD avoids this by invoking the LLM only for suspicious candidates. Across D1-D4, only 0.5% of windows require LLM validation on average. As shown in Table IV, compression also reduces the per-call input from 4100 tokens in LLMAD to 1100 tokens in CoLMAD, and the estimated daily cost decreases from 87.92 USD to 0.08 USD. The average LLM response time is 3.5 seconds per invoked candidate, but because invocation is sparse, the always-on monitoring path remains lightweight.

We also assess explanation usefulness through human evaluation. Compared with a standard prompt, anomaly-aware prompting improves usefulness by 13.4% and readability by 3.5%. The high satisfaction ratios for usefulness and readability are 72.37% and 97.11%, respectively. These results suggest that CoLMAD explanations are not merely auxiliary text; they provide structured evidence for manual verification.

Additional experiments show that the framework is not tied to a single LLM: GPT-4o delivers the most stable results, while DeepSeek-R1 remains competitive with GPT-3.5. Ablation results further show that removing structure-preserving compression, step-wise reasoning, or explicit normal/anomaly definitions causes the largest performance drops.

V. Case Studies and Lessons Learned

Fig. 3 illustrates how CoLMAD supports operational decision-making beyond producing an anomaly label.

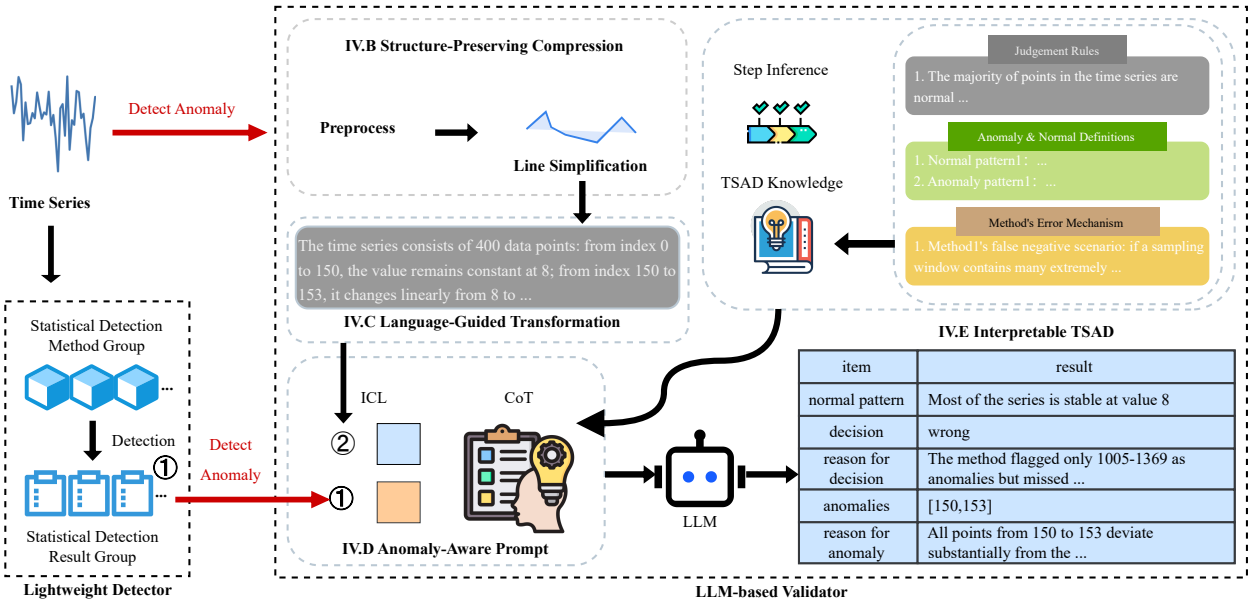

Fig. 2. Overview of CoLMAD. Lightweight detectors continuously scan incoming windows and generate suspicious candidates. The LLM validator is selectively invoked to compress, describe, reason about, and explain candidate anomalies.

TABLE II
DETECTION EFFECTIVENESS ACROSS PUBLIC AND PRODUCTION DATASETS. BEST RESULTS ARE IN BOLD AND SECOND-BEST RESULTS ARE UNDERLINED.

Method	D1			D2			D3			D4		
	Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1
CoLMAD	**0.958**	**1.000**	**0.979**	**0.974**	**1.000**	**0.987**	**0.979**	**1.000**	**0.989**	**0.913**	**1.000**	**0.955**
LOF	0.017	0.957	0.023	0.738	1.000	0.835	0.499	0.968	0.593	0.303	1.000	0.465
iForest	0.012	0.590	0.022	0.560	1.000	0.714	0.385	0.944	0.525	0.272	0.979	0.425
OCSVM	0.006	0.890	0.012	0.173	1.000	0.255	0.175	0.938	0.283	0.096	0.878	0.173
PUAD	0.260	0.862	0.400	0.770	1.000	0.870	0.698	0.974	0.813	0.611	0.949	0.744
Bagel	0.777	0.617	0.571	0.281	0.951	0.411	0.439	0.828	0.512	0.485	0.871	0.624
AnoTransfer	0.034	0.957	0.066	0.913	1.000	0.955	0.868	1.000	0.929	0.012	1.000	0.024
Donut	0.903	0.671	0.648	0.148	1.000	0.247	0.368	0.968	0.461	0.531	0.833	0.649
TimesNet	0.824	0.853	0.796	0.955	0.845	0.892	0.920	0.774	0.830	0.873	0.714	0.786
Informer	0.768	0.884	0.744	0.953	0.856	0.900	0.853	0.806	0.811	0.755	0.826	0.789
Timer zero-shot	0.708	0.824	0.688	0.963	0.851	0.900	0.912	0.781	0.829	0.818	0.874	0.845
Timer finetune	0.858	0.893	0.859	0.973	0.834	0.895	0.913	0.777	0.828	0.818	0.874	0.845
Moirai zero-shot	0.004	0.090	0.008	0.497	0.909	0.643	0.462	0.707	0.559	0.365	0.637	0.465
Moirai finetune	0.004	0.091	0.008	0.498	0.909	0.643	0.472	0.732	0.574	0.371	0.747	0.496
GPT4TS	0.987	0.034	0.066	0.985	0.533	0.674	0.915	0.426	0.543	0.899	0.199	0.325
LLMAD	0.733	0.671	0.700	0.642	0.707	0.673	0.651	0.683	0.666	0.684	0.712	0.697

TABLE III
DATASET STATISTICS.

	D1	D2	D3	D4
Interval (min.)	60	5	5	1
Total points	142100	158631	67740	589248
Anomaly points	466	15651	6316	31662
Anomaly ratio	0.328%	9.866%	9.324%	5.698%
Type	synthetic	real	real	production

TABLE IV
DEPLOYMENT-ORIENTED COMPARISON WITH THE DIRECT LLM-BASED BASELINE.

Item	LLMAD	CoLMAD
Input tokens / call	4100	1100
Output tokens / call	280	181
LLM invocation ratio	100%	0.5%
Avg. LLM latency / call	13.8s	3.5s
Daily cost (USD)	87.92	0.08

Case 1: local spike. The metric is mostly stable around 30, but an abrupt upward spike appears in $[480, 563]$. A lightweight statistical detector misses the segment because the abnormal behavior is local and short-lived. CoLMAD preserves the abrupt structural change during compression, describes it as a significant increase, and compares it with the normal stable pattern. The report identifies the anomalous interval and explains the reason as a spike. For an engineer, this is more useful than a binary label because it narrows the inspection scope and explains why the upstream detector failed.

Case 2: long-term shift. The second case contains a

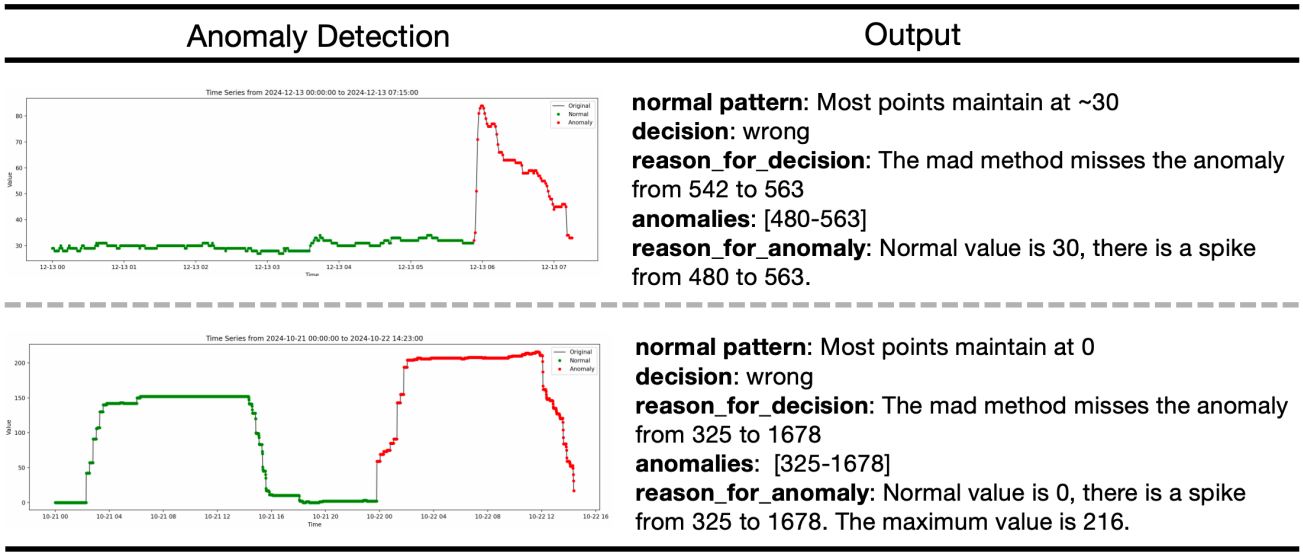

Fig. 3. Representative production-style cases. CoLMAD corrects detector errors and produces structured explanations for the anomalous interval and its cause.

sustained distribution shift: the series remains near 0 in the first half and then rises to a high plateau in $[325, 1678]$, with a peak value of 216. Detectors focused on isolated outliers can fail because the abnormal plateau contains many internally consistent values. CoLMAD captures the transition from the low-value stable region to the high-value plateau and marks the whole interval as anomalous due to persistent violation of the normal baseline. The explanation also records the maximum value, which helps operators prioritize the alert.

Operational implications. These cases illustrate how CoLMAD fits into an engineering workflow. The validator does not only return "anomaly" or "normal"; it returns a compact record that can be attached to an alert: the expected normal pattern, whether the upstream detector appears reliable, the anomalous interval, and the reason for the final decision. Such records help engineers narrow the inspection window, compare similar alerts, and decide whether a signal should be escalated to downstream diagnosis. They are also useful for postmortem analysis because the decision evidence is preserved together with the alert, rather than being reconstructed from raw dashboards after the incident.

Lesson 1: LLMs are most practical as selective validators. Our results and cases show that LLMs should not sit on the hot path for every window. The lightweight detector should provide broad coverage, while the LLM should be invoked only when semantic validation, false-alarm suppression, or explanation generation justifies the cost. This separation also provides a clear fallback: if the LLM service is unavailable, the system can still emit the lightweight detector result and mark the explanation as unavailable.

Lesson 2: Compression must preserve anomaly semantics. Token reduction alone is not sufficient. Aggressive downsampling can remove spikes, shift boundaries, or recovery trends that determine whether an interval is anomalous. In practice, compression should be evaluated by whether it preserves engineer-relevant evidence, such as turning points, abrupt jumps, plateau ranges, and return-to-normal segments.

Lesson 3: Explanations should be structured for action. Free-form explanations are hard to compare and audit. We found that engineers need stable fields: normal pattern, detector correctness, anomalous interval, reason for anomaly, and confidence-relevant evidence. These fields make LLM outputs easier to display in alert dashboards, store with incident records, and pass to downstream diagnosis tools.

VI. Deployment Considerations

A practical deployment of CoLMAD should treat the LLM validator as a refinement layer rather than a dependency of the alerting path. If the LLM service is slow or unavailable, the monitoring system can still emit the lightweight detector result and mark the explanation as unavailable. This fallback keeps the reliability path predictable while allowing operators to benefit from LLM-based reasoning when the service is healthy.

The current framework focuses on univariate metric streams. This setting is common in production dashboards and allows clear interval-level explanations, but many incidents involve correlations across metrics, logs, traces, and deployment events. Extending CoLMAD to multi-source evidence is promising, but the additional context must be selected carefully; otherwise, the LLM may receive irrelevant information and lose the compactness that makes the validator cost-effective.

Another practical consideration is knowledge maintenance. The anomaly-aware prompt contains normality definitions, anomaly definitions, and detector error mechanisms. In production, these rules should be versioned like operational knowledge. New rule blocks can be added for recurring metric families, such as latency spikes, throughput drops, utilization saturation, and periodic workload changes. This keeps the

prompt reusable while allowing domain adaptation without retraining the lightweight detector.

VII. Conclusion

We presented CoLMAD, an LLM-assisted TSAD framework designed for industrial monitoring. By combining always-on lightweight detectors with selective, structure-aware LLM validation, CoLMAD achieves strong accuracy, low invocation cost, and interpretable anomaly reports on both public and production datasets. These results show that LLMs can be practical for TSAD when used as controlled validators rather than full replacements for scalable detectors.

References

[1] D. Liu, Y. Zhao, H. Xu, Y. Sun, D. Pei, J. Luo, X. Jing, and M. Feng, "Opprentice: Towards practical and automatic anomaly detection through machine learning," in *Proceedings of the 2015 internet measurement conference*, 2015, pp. 211–224.

[2] S. Zhang, Y. Liu, D. Pei, Y. Chen, X. Qu, S. Tao, and Z. Zang, "Rapid and robust impact assessment of software changes in large internet-based services," in *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*, 2015, pp. 1–13.

[3] S. Zhang, Y. Liu, D. Pei, Y. Chen, X. Qu, S. Tao, Z. Zang, X. Jing, and M. Feng, "Funnel: Assessing software changes in web-based services," *IEEE Transactions on Services Computing*, vol. 11, no. 1, pp. 34–48, 2016.

[4] S. Zhang, C. Zhao, Y. Sui, Y. Su, Y. Sun, Y. Zhang, D. Pei, and Y. Wang, "Robust kpi anomaly detection for large-scale software services with partial labels," in *2021 IEEE 32nd international symposium on software reliability engineering (ISSRE)*. IEEE, 2021, pp. 103–114.

[5] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[6] G. E. Box and D. A. Pierce, "Distribution of residual autocorrelations in autoregressive-integrated moving average time series models," *Journal of the American statistical Association*, vol. 65, no. 332, pp. 1509–1526, 1970.

[7] H. Hotelling *et al.*, "The generalization of student's ratio," 1931.

[8] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 2000, pp. 93–104.

[9] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 eighth ieee international conference on data mining*. IEEE, 2008, pp. 413–422.

[10] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM computing surveys (CSUR)*, vol. 46, no. 4, pp. 1–37, 2014.

[11] M. Ma, S. Zhang, D. Pei, X. Huang, and H. Dai, "Robust and rapid adaption for concept drift in software system anomaly detection," in *2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2018, pp. 13–24.

[12] S. Schmidl, P. Wenig, and T. Papenbrock, "Anomaly detection in time series: a comprehensive evaluation," *Proceedings of the VLDB Endowment*, vol. 15, no. 9, pp. 1779–1797, 2022.

[13] Z. Li, W. Chen, and D. Pei, "Robust and unsupervised kpi anomaly detection based on conditional variational autoencoder," in *2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC)*. IEEE, 2018, pp. 1–9.

[14] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng *et al.*, "Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications," in *Proceedings of the 2018 world wide web conference*, 2018, pp. 187–196.

[15] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 2828–2837.

[16] J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M. A. Zuluaga, "Usad: Unsupervised anomaly detection on multivariate time series," in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020, pp. 3395–3404.

[17] A. Geiger, D. Liu, S. Alnegheimish, A. Cuesta-Infante, and K. Veeramachaneni, "Tadgan: Time series anomaly detection using generative adversarial networks," in *2020 ieee international conference on big data (big data)*. IEEE, 2020, pp. 33–43.

[18] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, and M. Long, "Timesnet: Temporal 2d-variation modeling for general time series analysis," *arXiv preprint arXiv:2210.02186*, 2022.

[19] Y. Liu, H. Zhang, C. Li, X. Huang, J. Wang, and M. Long, "Timer: Generative pre-trained transformers are large time series models," *arXiv preprint arXiv:2402.02368*, 2024.

[20] M. Goswami, K. Szafer, A. Choudhry, Y. Cai, S. Li, and A. Dubrawski, "Moment: A family of open time-series foundation models," *arXiv preprint arXiv:2402.03885*, 2024.

[21] G. Woo, C. Liu, A. Kumar, C. Xiong, S. Savarese, and D. Sahoo, "Unified training of universal time series forecasting transformers," 2024.

[22] T. Zhou, P. Niu, L. Sun, R. Jin *et al.*, "One fits all: Power general time series analysis by pretrained lm," *Advances in neural information processing systems*, vol. 36, pp. 43 322–43 355, 2023.

[23] N. Gruver, M. Finzi, S. Qiu, and A. G. Wilson, "Large language models are zero-shot time series forecasters," *Advances in Neural Information Processing Systems*, vol. 36, pp. 19 622–19 635, 2023.

[24] M. Jin, S. Wang, L. Ma, Z. Chu, J. Y. Zhang, X. Shi, P.-Y. Chen, Y. Liang, Y.-F. Li, S. Pan *et al.*, "Time-llm: Time series forecasting by reprogramming large language models," *arXiv preprint arXiv:2310.01728*, 2023.

[25] J. Liu, C. Zhang, J. Qian, M. Ma, S. Qin, C. Bansal, Q. Lin, S. Rajmohan, and D. Zhang, "Large language models can deliver accurate and interpretable time series anomaly detection," *arXiv preprint arXiv:2405.15370*, 2024.

[26] S. Alnegheimish, L. Nguyen, L. Berti-Equille, and K. Veeramachaneni, "Large language models can be zero-shot anomaly detectors for time series?" *arXiv preprint arXiv:2405.14755*, 2024.

[27] J. Su, C. Jiang, X. Jin, Y. Qiao, T. Xiao, H. Ma, R. Wei, Z. Jing, J. Xu, and J. Lin, "Large language models for forecasting and anomaly detection: A systematic literature review," *arXiv preprint arXiv:2402.10350*, 2024.

[28] N. Laptev, S. Amizadeh, and I. Flint, "Generic and scalable framework for automated time-series anomaly detection," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 1939–1947.

[29] A. Lavin and S. Ahmad, "Evaluating real-time anomaly detection algorithms–the numenta anomaly benchmark," in *2015 IEEE 14th international conference on machine learning and applications (ICMLA)*. IEEE, 2015, pp. 38–44.

[30] J. Ma and S. Perkins, "Time-series novelty detection using one-class support vector machines," in *Proceedings of the International Joint Conference on Neural Networks, 2003.*, vol. 3. IEEE, 2003, pp. 1741–1745.

[31] S. Zhang, Z. Zhong, D. Li, Q. Fan, Y. Sun, M. Zhu, Y. Zhang, D. Pei, J. Sun, Y. Liu *et al.*, "Efficient kpi anomaly detection through transfer learning for large-scale web services," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 8, pp. 2440–2455, 2022.

[32] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 12, 2021, pp. 11 106–11 115.