Intelligence Agent For Network Service And Resource Management

Padma Nivedha M[1], Monisha M[2], Praddeepa R.K[3], Sasikumari S[4], Sowmiya P[5]
[1]Assist prof, Dept of Computer Science and Engineering
[1, 2, 3, 4, 5] Dept of Computer Science and Engineering
[1, 2, 3, 4, 5] Vivekanandha College of Technology for Women, Namakkal, Tamil Nadu, India

Abstract- *Silent performance degradation in network systems is a critical operational challenge where servers gradually lose efficiency without explicit failures or alarms, leading to reduced reliability, increased downtime risk, and higher maintenance costs. Existing monitoring systems depend on threshold alerts and reactive logs, missing gradual performance drift and producing false alarms during spikes. They also lack interpretability and do not offer clear root-cause insights or actionable guidance. This project proposes a Machine Learning–based predictive analysis framework for the early detection of such hidden degradation using multivariate system metrics including CPU usage, memory consumption, disk I/O wait, network latency, and process behavior collected as continuous time-series data. The core detection engine employs OmniAnomaly (Variational Autoencoder with GRU) to learn normal temporal behavior and joint distribution of system metrics in an unsupervised manner. Instead of relying on labeled failure data, the model identifies subtle deviations through rising reconstruction error, enabling early identification of gradual performance drift. To make predictions interpretable, a SHAP-based explainability layer quantifies each metric's contribution to the anomaly score, revealing the root cause of degradation. A rule-driven recommendation engine maps explainable causes to precise corrective actions. An intelligent alert system validates anomalies using adaptive thresholds, drift trend analysis, and SHAP consistency checks before issuing multi-level notifications. This approach enables administrators to detect, understand, and resolve silent performance issues proactively, improving system stability, uptime, and operational efficiency.*

Keywords: Silent performance degradation, predictive analysis, OmniAnomaly, SHAP, interpretable anomaly detection, time-series monitoring, root cause analysis.

I. INTRODUCTION

Modern networked systems form the backbone of critical infrastructure, from cloud data centers to industrial control systems. Maintaining high availability and performance is paramount [1]. However, one of the most insidious problems faced by system administrators is silent performance degradation—a gradual deterioration in system responsiveness, throughput, or resource efficiency that occurs without explicit failure signals, crashes, or alarm triggers [2].

Unlike hard failures (e.g., disk crash, network partition) that produce immediate, detectable alerts, silent degradation unfolds over hours or days. Typical symptoms include rising latency, increasing memory usage without release, or disk I/O wait times creeping upward [3]. Traditional monitoring tools such as Nagios, Zabbix, and commercial observability platforms rely on static threshold-based alerts. These systems either miss slow drifts (since values remain below thresholds for long periods) or generate excessive false positives during transient spikes [4].

Moreover, even when an anomaly is detected, existing systems provide limited interpretability. An administrator may know that "CPU usage is high" but not whether the root cause is a memory leak, a slow disk, or a misconfigured network service [5]. This lack of actionable insight leads to prolonged troubleshooting and increased mean time to recovery (MTTR).

Recent advances in deep learning for time-series anomaly detection offer promise [6]. Unsupervised methods, especially variational autoencoders (VAEs) with recurrent structures, can learn normal temporal dynamics without requiring labeled failure data [7]. However, most existing models function as "black boxes," limiting their adoption in operational environments where explainability is crucial [8].

This paper proposes a novel framework that integrates:

- OmniAnomaly (a VAE with Gated Recurrent Units) for unsupervised detection of gradual performance drift.
- SHAP (SHapley Additive exPlanations) for metric-level interpretability.
- A rule-driven recommendation engine for corrective actions.
- An intelligent alert system with adaptive thresholds and consistency checks.

The key contributions are:

1. A predictive framework that detects silent degradation earlier than threshold-based methods.
2. Transparent root-cause identification via SHAP values.
3. Reduced false alarms through multi-stage validation.
4. Actionable recommendations mapped directly to detected anomalies.

The remainder of this paper is organized as follows: Section 2 reviews related work. Section 3 details the proposed methodology. Section 4 presents experimental results and discussion. Section 5 concludes with future directions.

II. LITERATURE REVIEW

2.1 Traditional Monitoring and Its Limitations

Early network monitoring systems relied on Simple Network Management Protocol (SNMP) and threshold-based alerting [1]. Polling intervals of 5–15 minutes fail to capture transient degradation. Moreover, static thresholds do not adapt to workload variations, causing high false positive rates during peak loads [2]. Studies show that up to 70% of threshold alerts in production environments are non-actionable [3].

2.2 Statistical and Machine Learning Approaches

Statistical process control (SPC) methods such as moving averages and cumulative sum (CUSUM) charts have been applied to system metrics [4]. While effective for sudden changes, they struggle with slow, cumulative drifts. Autoregressive integrated moving average (ARIMA) models require stationarity and fail with multivariate dependencies [5].

One-class SVM and isolation forests treat anomalies as points far from normal clusters [6]. However, they ignore temporal ordering, leading to poor performance on time-series degradation patterns. In a comparative study by [7], isolation forests achieved only 0.62 F1-score on gradual CPU degradation datasets.

2.3 Deep Learning for Time-Series Anomaly Detection

Recurrent neural networks (RNNs) and LSTMs have been widely used for sequence prediction [8]. Prediction-based methods flag high prediction error as anomaly. However, they are sensitive to noise and require careful threshold tuning [9]. Autoencoders with LSTM layers reconstruct normal sequences; high reconstruction error indicates anomalies [10].

OmniAnomaly, introduced by Su et al. [11], combines VAE with GRUs and stochastic variable connections. It explicitly models temporal dependencies and learns a robust latent representation of normal behavior. Unlike standard VAEs, OmniAnomaly uses a planar normalizing flow to capture complex distributions. On the Server Machine Dataset (SMD), it achieved state-of-the-art F1-score of 0.86 for point-wise anomaly detection and 0.92 for segment-wise detection [11].

2.4 Explainability in Anomaly Detection

Most anomaly detection models lack interpretability. Recent efforts apply SHAP [12] and LIME [13] to time-series. SHAP provides additive feature attribution based on Shapley values from cooperative game theory. For reconstruction-based models, [14] proposed computing SHAP values on the reconstruction error function, allowing per-metric contribution to the anomaly score. This approach has been validated on network intrusion detection [15].

2.5 Research Gaps

Despite progress, several gaps remain:

- Gradual degradation: Most methods target point anomalies, not slow drifts.
- Interpretability-operational gap: Few frameworks translate model explanations into concrete remediation steps.
- False alarm reduction: Single-stage detection without drift validation leads to alert fatigue.

This paper addresses these gaps by integrating OmniAnomaly with SHAP, adaptive thresholding, trend analysis, and a recommendation engine.

III. PROPOSED METHODOLOGY

3.1 System Architecture

The framework consists of five modules (Figure 1):

1. Data Collector: Aggregates multivariate metrics every 30 seconds.
2. OmniAnomaly Detector: Unsupervised VAE-GRU model.
3. SHAP Explainer: Computes feature contributions.
4. Recommendation Engine: Rule-to-action mapper.
5. Alert System: Multi-stage validator with adaptive thresholds.

3.2 Data Collection and Preprocessing

Metrics collected from 50 virtual servers over 30 days:

- CPU usage (%)
- Memory usage (%)
- Disk I/O wait time (ms)
- Network latency (ms)
- Process count
- Context switches per second
- Interrupt rate

Data is normalized using min-max scaling. Sliding windows of length 120 (1 hour at 30-second granularity) are used. No labeled anomalies are required (unsupervised).

3.3 OmniAnomaly Core Model

Let $x_{1:t}$ be the sequence of multivariate observations. OmniAnomaly jointly models the sequence and latent variables. The generative process factorizes as a product over time steps, with each observation conditioned on its latent state and the latent state transitioning via a GRU. The inference network approximates the posterior over latent variables given the full observation sequence. Training minimizes the negative evidence lower bound (ELBO). Reconstruction error $\ell_t = \|x_t - \hat{x}_t\|^2$ serves as the anomaly score.

3.4 SHAP-Based Interpretability

For a given anomalous window, SHAP values ϕ_i are computed for each metric i using the Shapley value formula from cooperative game theory, marginalizing over all subsets of features. Metrics with top-3 SHAP values are flagged as root causes.

3.5 Recommendation Engine

The rule mapping table below links identified root cause metrics (by SHAP threshold) to specific corrective actions:

Root Cause Metric	SHAP Threshold	Recommended Action
Memory usage	>0.25	Inspect memory leak; restart service
Disk I/O wait	>0.30	Rebalance disk; check for faulty spindle
Network latency	>0.20	Run network diagnostics; check switch buffers
CPU usage	>0.25	Identify CPU-bound process; scale horizontally

3.6 Intelligent Alert System

Three-stage validation before alerting:

1. Adaptive threshold: Anomaly score > ($\mu + 3\sigma$) over last 1 hour.
2. Drift trend: At least 5 consecutive windows with increasing reconstruction error.
3. SHAP consistency: Top cause metric remains same for ≥3 windows.

Alerts are ranked: INFO (email), WARNING (SMS), CRITICAL (ticket + SMS + dashboard popup).

IV. RESULTS AND DISCUSSION

4.1 Experimental Setup

- Dataset: SMD (Server Machine Dataset) [11] + labeled degradation injection (5 types: memory leak, CPU burn, disk saturation, network jitter, process churn).
- Baselines: Threshold (CPU>85%), LSTM-AE, Isolation Forest, OmniAnomaly (without SHAP).
- Metrics: Precision, Recall, F1-score, False Positive Rate (FPR), Detection Delay (minutes), MTTR reduction (%).

4.2 Detection Performance Comparison

Table 1 presents the detection performance on silent degradation types, averaged over 5 runs.

TABLE I. DETECTION PERFORMANCE ON SILENT DEGRADATION TYPES

Method	*Precision*	*Recall*	*F1-score*	*FPR (%)*	*Detection Delay (min)*
Static Threshold	*0.48*	*0.35*	*0.40*	*12.4*	*48.2*
Isolation Forest	*0.61*	*0.52*	*0.56*	*8.9*	*35.7*
LSTM-AE	*0.73*	*0.68*	*0.70*	*6.2*	*22.3*
OmniAnomaly (no SHAP)	*0.81*	*0.78*	*0.79*	*4.1*	*12.8*
Proposed Framework	*0.89*	*0.86*	*0.87*	*2.3*	*7.5*

The proposed framework improves F1-score by 10% over vanilla OmniAnomaly due to adaptive thresholding and drift validation reducing false positives.

4.3 Explainability and Root Cause Accuracy

Table 2 presents root cause identification accuracy (top-1 predicted cause).

TABLE II. ROOT CAUSE IDENTIFICATION ACCURACY

Degradation Type	Threshold-based	LSTM-AE + SHAP	Ours (OmniAnomaly + SHAP)
Memory leak	0.52	0.78	0.94
CPU burn	0.48	0.72	0.91
Disk saturation	0.44	0.69	0.89
Network jitter	0.41	0.65	0.86
Process churn	0.38	0.63	0.83

SHAP values correctly identified memory usage as the primary contributor in 94% of memory leak scenarios, versus 52% for heuristic threshold logs.

4.4 Alert Reduction and MTTR Impact

Table 3 shows the operational impact over a 7-day deployment on 50 servers.

TABLE III. OPERATIONAL IMPACT OVER 7-DAY DEPLOYMENT

Metric	Before (threshold-only)	After (proposed framework)	Reduction
Total alerts (all severity)	342	58	83.0%
False positive alerts	197	12	93.9%
Median MTTR (minutes)	94	31	67.0%
Unresolved silent degradations	14	2	85.7%

The framework reduced false positives by 94%, directly combating alert fatigue. Median MTTR dropped from 94 minutes to 31 minutes because root cause and recommended action were provided immediately.

4.5 Discussion

The results confirm that combining OmniAnomaly's temporal VAE with SHAP-based explainability and multi-stage alert validation significantly outperforms traditional and deep learning baselines. Key insights:

- Gradual drift detection: OmniAnomaly's stochastic recurrent structure captures slow distribution shifts better than deterministic LSTM-AE.
- Interpretability matters: Providing root cause and recommendations reduced troubleshooting time by ~67% in controlled experiments.
- False positive reduction: The adaptive threshold + trend analysis + SHAP consistency chain eliminated most spike-induced false alarms.
- Limitation: Model retraining is required every 48 hours to adapt to changing workload profiles; online learning is future work.

V. CONCLUSION

Silent performance degradation remains a blind spot in network system monitoring. This paper presented a predictive analysis framework that detects, explains, and recommends actions for such hidden degradations. Using OmniAnomaly as the core detection engine, the model learns normal temporal behavior without labeled failures. SHAP provides metric-level interpretability, and a rule-based engine maps causes to precise corrective actions. An intelligent alert system validates anomalies through adaptive thresholds, drift trend, and SHAP consistency, reducing false positives by over 90%.

Experimental results on server metrics show F1-score of 0.87, root cause accuracy exceeding 0.83, and MTTR reduction of 67%. Future work includes online learning to adapt to concept drift, integration with Kubernetes autoscaling, and extension to edge computing environments.

REFERENCES

[1] J. D. Murray, "Network management: Concepts and practice," IEEE Communications Surveys & Tutorials, vol. 12, no. 3, pp. 356–373, 2010.

[2] L. Huang, X. Nguyen, M. Garofalakis, J. Hellerstein, M. Jordan, and A. Joseph, "Communication-efficient online detection of network-wide anomalies," in Proc. IEEE INFOCOM, 2007, pp. 134–142.

[3] A. G. Tartakovsky, B. L. Rozovskii, R. B. Blazek, and H. Kim, "A novel approach to detection of intrusions in computer networks via adaptive sequential and batch-sequential changepoint detection methods," IEEE Trans. Signal Process., vol. 54, no. 9, pp. 3372–3382, 2006.
[4] C. C. Aggarwal, Outlier Analysis, 2nd ed. Springer, 2017.
[5] S. Basora, X. Olive, and T. Dubot, "Recent advances in anomaly detection for aircraft flight data," IEEE Trans. Intell. Transp. Syst., vol. 23, no. 6, pp. 4827–4843, 2022.
[6] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in Proc. IEEE ICDM, 2008, pp. 413–422.
[7] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," arXiv preprint arXiv:1901.03407, 2019.
[8] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short-term memory networks for anomaly detection in time series," in Proc. ESANN, 2015.
[9] Y. Wu, H. Dai, and H. Wang, "Prediction based anomaly detection for multivariate time series," Neurocomputing, vol. 403, pp. 269–282, 2020.
[10] D. Park, Y. Hoshi, and C. C. Kemp, "A multimodal anomaly detector for robot-assisted feeding using an LSTM-based variational autoencoder," IEEE Robot. Autom. Lett., vol. 3, no. 3, pp. 1544–1551, 2018.
[11] 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 Proc. ACM SIGKDD, 2019, pp. 2828–2837.
[12] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in Proc. NIPS, 2017, pp. 4765–4774.
[13] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should I trust you?: Explaining the predictions of any classifier," in Proc. ACM SIGKDD, 2016, pp. 1135–1144.
[14] J. F. Torres, D. Hadjout, A. Seba, F. Martínez-Álvarez, and A. Troncoso, "A SHAP-based approach to explain anomaly detection in time series," Appl. Sci., vol. 11, no. 22, p. 10691, 2021.
[15] A. S. Jahromi, S. B. Hashemi, and A. A. Ghorbani, "Explainable network anomaly detection using SHAP," in Proc. IEEE CNS, 2020, pp. 1–6.