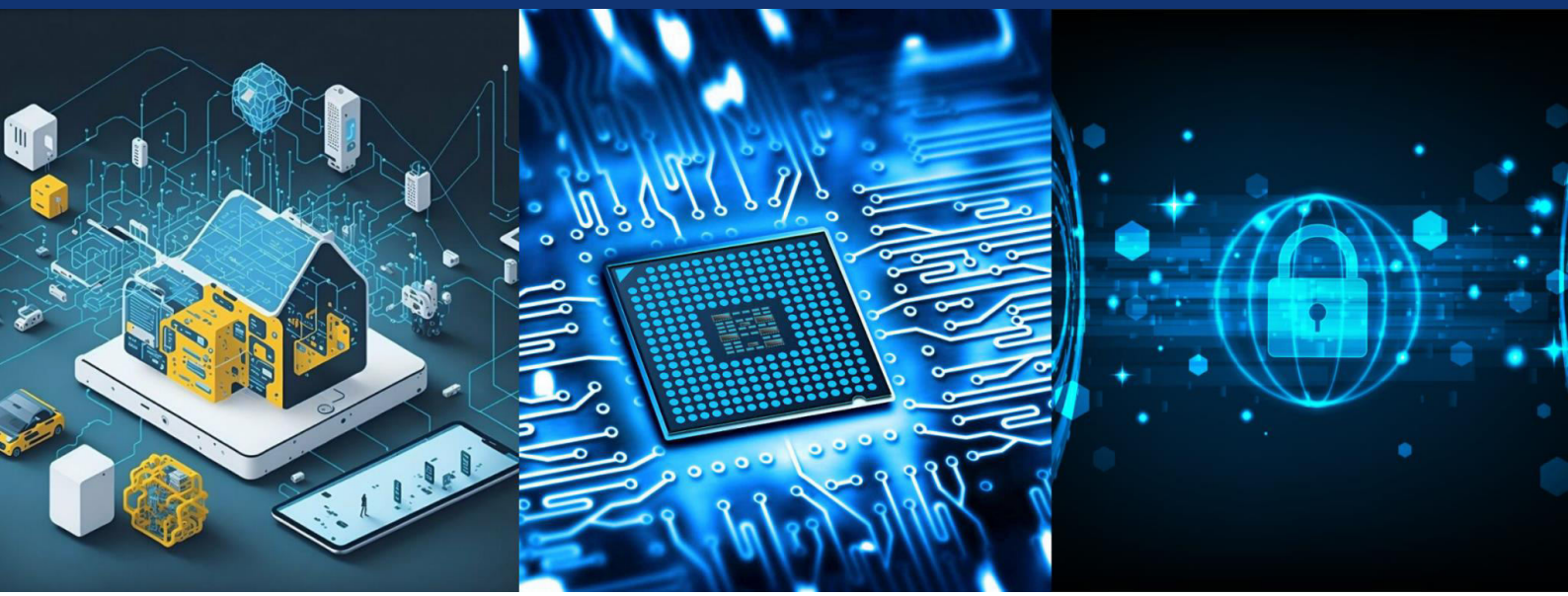
IJIRCCE

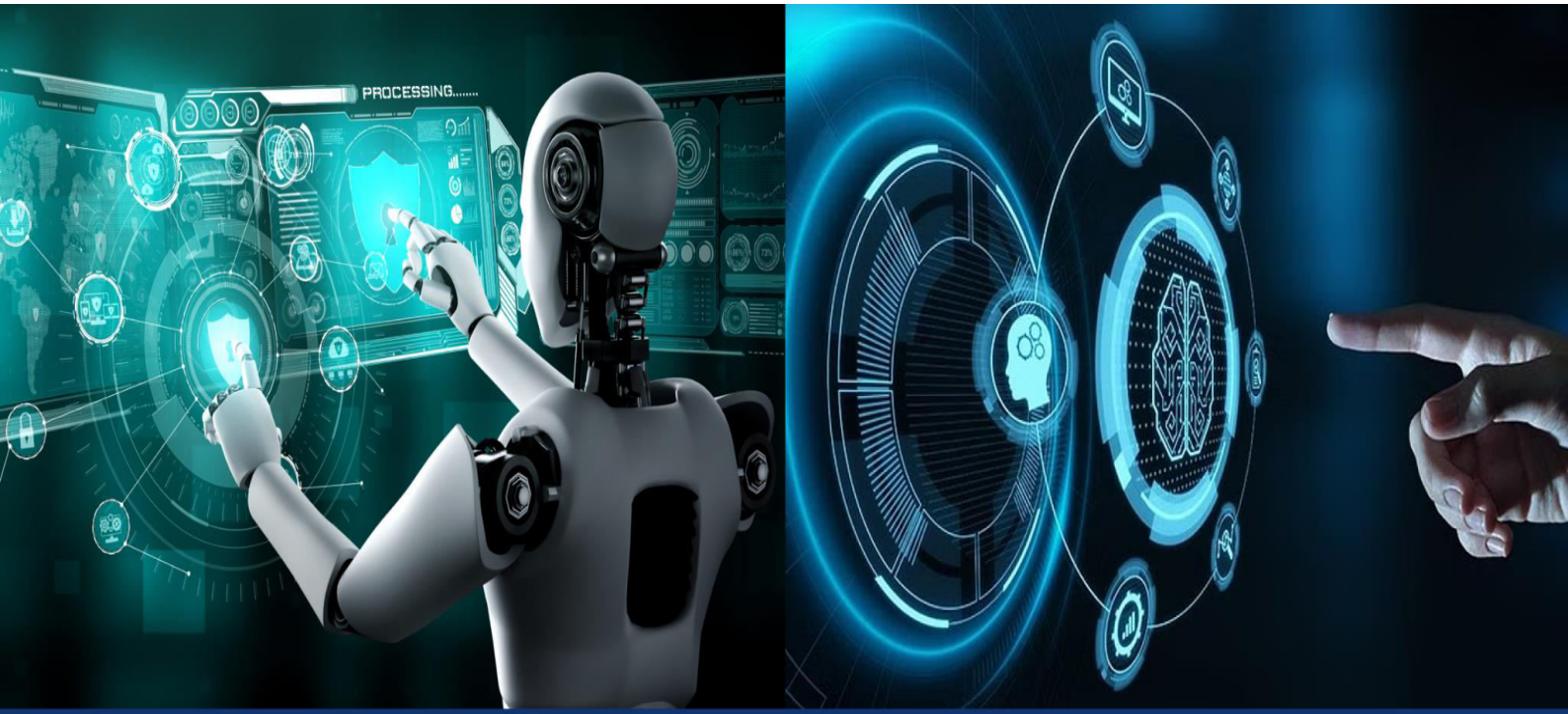
ISSN(O): 2320-9801

ISSN(P): 2320-9798



International Journal of Innovative Research in Computer and Communication Engineering

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 9.927

Volume 14, Issue 5, May 2026

© 2026 IJIRCCE | Volume 14, Issue 5, May 2026 DOI: 10.15680/IJIRCCE.2026.1405063

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

AI-Driven CI/CD Platform Reliability: A Framework for Intelligent Fault Detection, Predictive Maintenance, and Automated Recovery

Swapneel G[1], Sujan M A[2], Siddaraju H S[3], Vishwanath B S[4], Talakal Krushna[5], Vinod D[6], Prof. Manjula P[7]

UG Students, Dept. of CSE, Jain Institute of Technology, Davangere, Karnataka, India[1,2,3,4,5,6]

Assistant Professor, Dept. of CSE, Jain Institute of Technology, Davangere, Karnataka, India[7]

ABSTRACT: Continuous Integration and Continuous Delivery (CI/CD) pipelines are the operational backbone of modern software delivery. As organizations scale these pipelines to support hundreds of microservices and thousands of daily deployments, ensuring platform reliability becomes a critical engineering challenge. This paper presents a framework for AI-Driven CI/CD Platform Reliability that integrates intelligent fault detection, predictive maintenance, and automated recovery mechanisms. The proposed system employs machine learning models trained on historical pipeline telemetry, build logs, and infrastructure metrics to identify anomalies before they cause service disruption. A predictive maintenance module anticipates component degradation and schedules preventive actions, while an automated recovery engine applies validated remediation playbooks in response to detected faults. Experimental evaluation on simulated and production-trace datasets demonstrates significant reductions in mean time to recovery (MTTR), pipeline failure rates, and manual operator interventions. The framework advances the state of intelligent DevOps by combining anomaly detection, time-series forecasting, and reinforcement-learning-based self-healing into a unified, extensible platform.

KEYWORDS: CI/CD Reliability, Fault Detection, Predictive Maintenance, Automated Recovery, DevOps Intelligence, Anomaly Detection, Self-Healing Systems, MLOps, Pipeline Observability.

I. INTRODUCTION

Modern software organizations rely on Continuous Integration and Continuous Delivery (CI/CD) pipelines to deliver features rapidly and reliably. A CI/CD pipeline orchestrates code compilation, automated testing, security scanning, artifact packaging, and deployment across complex, distributed infrastructure. When a pipeline fails, the downstream impact spans delayed releases, degraded customer experience, and lost engineering productivity.

Despite their centrality, CI/CD platforms remain brittle under scale. Flaky tests, resource contention, network transients, dependency version drift, and infrastructure brownouts collectively cause a non-trivial fraction of pipeline failures in enterprise environments. Traditional approaches to reliability rely on reactive monitoring: operators receive alerts after a failure manifests, investigate logs manually, and apply fixes on an ad hoc basis. This reactive paradigm is insufficient at the pace and scale of modern deployments.

This paper proposes an AI-Driven CI/CD Reliability Framework that shifts the paradigm from reactive fire-fighting to proactive, intelligent platform management. The framework integrates three tightly coupled subsystems: (1) an intelligent fault detection engine that identifies anomalies in real time using stream processing and supervised machine learning; (2) a predictive maintenance module that forecasts component-level degradation using time-series models; and (3) an automated recovery engine that selects and executes remediation actions using a reinforcement learning policy trained on historical incident-resolution data.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 describes the proposed framework and its algorithms. Section 4 presents simulation results. Section 5 concludes with future directions.

© 2026 IJIRCCE | Volume 14, Issue 5, May 2026 DOI: 10.15680/IJIRCCE.2026.1405063

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

II. RELATED WORK

Research at the intersection of machine learning and software operations has grown rapidly. AIOps platforms leverage large-scale telemetry to automate incident management; early systems such as IBM Watson AIOps and Moogsoft demonstrated that log-based anomaly detection can reduce alert fatigue. However, these systems are generic observability tools and do not model the structural characteristics of CI/CD pipelines.

Flaky test detection has been studied extensively. Luo et al. identified rerun-based heuristics for detecting non-deterministic tests, while subsequent work applied natural language processing to test names and failure messages to improve recall. These approaches address a narrow fault class and do not generalize to infrastructure-level failures.

Predictive failure analysis in distributed systems has employed LSTM networks, Prophet models, and gradient-boosted regression on time-series metrics. Google's Site Reliability Engineering (SRE) literature advocates for error budgets and service level objectives (SLOs), yet does not prescribe ML-based automation for pipeline management specifically.

Automated remediation via runbook automation exists in tools such as PagerDuty and Opsgenie, but these rely on hand-crafted rules. Reinforcement learning for cloud resource management has shown promise in hyperparameter tuning and auto-scaling, yet applications to CI/CD fault recovery remain nascent. Our work fills this gap by unifying fault detection, predictive maintenance, and RL-driven self-healing within a single, CI/CD-aware framework.

III. PROPOSED FRAMEWORK

The AI-Driven CI/CD Reliability Framework comprises four layers: a telemetry ingestion layer, an intelligent fault detection module, a predictive maintenance module, and an automated recovery engine. Figure 3.1 illustrates the high-level architecture.

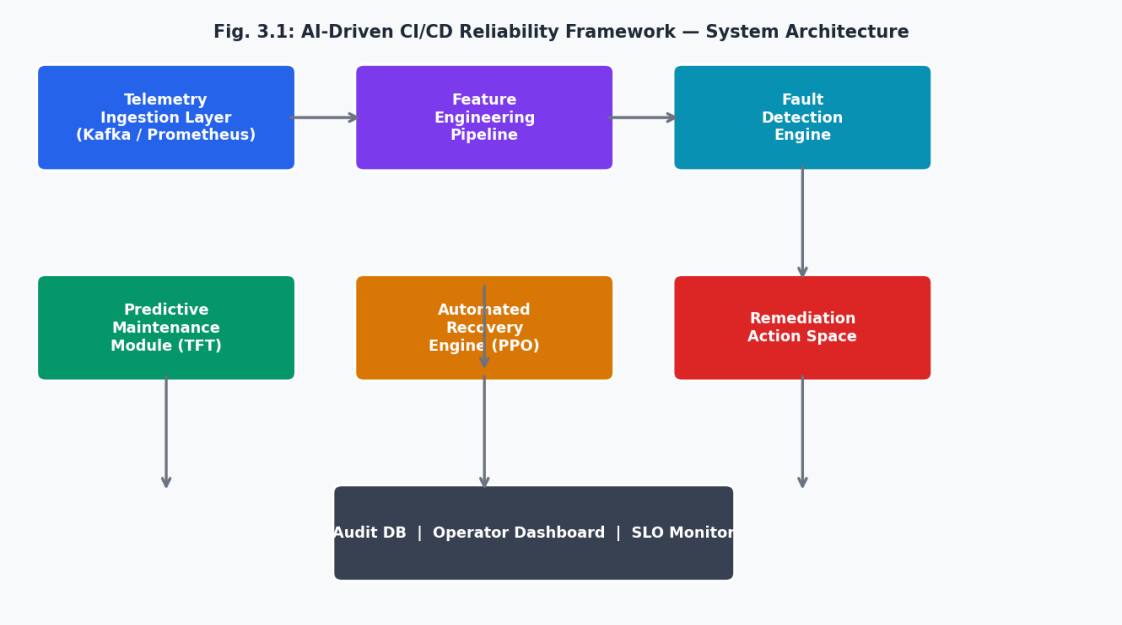


Fig. 3.1: AI-Driven CI/CD Reliability Framework — System Architecture

3.1 Telemetry Ingestion and Feature Engineering

Raw observability data is collected from three primary sources: (a) structured build logs emitted by CI runners, (b) infrastructure metrics (CPU utilization, memory pressure, disk I/O, network latency) scraped via Prometheus, and (c) pipeline event streams (job queued, job started, job failed, deployment succeeded) published to Apache Kafka topics.

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

A feature engineering pipeline transforms raw signals into model-ready tensors. Continuous metrics are windowed into fixed-size rolling buffers of width W = 60 seconds and aggregated to produce statistical features including mean, standard deviation, skewness, and percentile values. Log lines are tokenized and embedded using a fine-tuned BERT-based encoder, producing dense semantic representations. The composite feature vector for a pipeline run r at time t is:

$$F(r, t) = [M(t), L(t), E(t)]$$

Where M(t) denotes metric aggregates, L(t) denotes log embeddings, and E(t) denotes event-sequence encodings. This unified representation is consumed by downstream detection and forecasting modules.

3.2 Intelligent Fault Detection

The fault detection module processes the feature stream in near real time to classify each pipeline run as nominal or anomalous. Two complementary detection strategies are employed.

First, a supervised classifier trained on labeled historical incidents detects known failure patterns. The classifier is a gradient-boosted decision tree ensemble (XGBoost) trained on features extracted from 18 months of production pipeline telemetry. The binary classification function is:

$$P(\text{fault} \mid F(r, t)) = \text{Sigmoid}(\text{XGBoost}(F(r, t)))$$

Second, an unsupervised autoencoder captures the nominal distribution of pipeline behavior and flags deviations whose reconstruction error exceeds a learned threshold:

$$A(r, t) = ||F(r, t) - \text{Decoder}(\text{Encoder}(F(r, t)))||^2 > \theta$$

Where theta is calibrated on a held-out validation set to achieve a false positive rate below 5%. Alerts from both detectors are fused using a weighted majority vote, prioritizing precision for high-severity fault classes to minimize alert fatigue.

3.3 Predictive Maintenance Module

While fault detection reacts to imminent failures, the predictive maintenance module anticipates component degradation over longer horizons. It applies a Temporal Fusion Transformer (TFT) model to multivariate time series of infrastructure health indicators, forecasting the probability of component failure within a configurable lookahead window (default: 30 minutes).

The forecasting objective minimizes quantile loss across prediction horizons:

$$L = \sum_q \sum_t QL(q, y_t, \hat{y}_{q,t})$$

When the predicted failure probability exceeds a configurable threshold alpha (default: 0.75), the maintenance scheduler issues a preventive work order — draining a CI runner, triggering garbage collection, or rotating a degraded database connection pool.

3.4 Automated Recovery Engine

Upon confirmed fault detection, the recovery engine selects the optimal remediation action from a finite action space A = {retry_job, restart_runner, revert_deployment, scale_out_runner_pool, invalidate_cache, notify_operator}. The selection policy is learned via Proximal Policy Optimization (PPO), treating incident resolution as a Markov Decision Process (MDP). The reward function incentivizes fast recovery while penalizing side effects:

$$R = -\text{MTTR}_{\text{normalized}} + \lambda \cdot \text{Success\_flag} - \mu \cdot \text{Blast\_radius}$$

The trained policy generalizes across fault types not seen during training, falling back to a rule-based heuristic when confidence is below a safety threshold.

3.5 Pseudocode of Proposed Algorithm

- Initialize telemetry ingestion pipelines (Kafka consumers, Prometheus scrapers)
- Load pretrained XGBoost classifier, Autoencoder, TFT forecaster, and PPO policy
- For each incoming pipeline event stream:
- Extract and engineer feature vector F(r, t)
- Run fault detection: compute supervised score and reconstruction error
- If either detector exceeds threshold: raise fault alert

© 2026 IJIRCCE | Volume 14, Issue 5, May 2026 DOI: 10.15680/IJIRCCE.2026.1405063

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

- Run predictive maintenance: forecast component health for next 30 minutes
- If predicted failure probability > alpha: schedule preventive action
- If fault alert raised: invoke recovery engine; select action via PPO policy
- Execute selected remediation action; observe outcome
- Update policy replay buffer; retrain models on new incident data periodically
- Log all decisions and outcomes to audit database

IV. SIMULATION RESULTS

The framework was evaluated on two datasets: (1) a synthetic dataset of 50,000 simulated pipeline runs with injected faults spanning flaky tests, runner OOM events, network timeouts, and deployment rollback failures; and (2) a production trace dataset of 12,000 pipeline runs from a mid-sized software organization covering a 90-day period.

Fault Detection Performance: The supervised XGBoost classifier achieved a precision of 94.2% and recall of 91.7% on the synthetic dataset, with an AUC-ROC of 0.973. The autoencoder complemented the classifier by detecting 83% of novel fault patterns unseen in training, reducing missed detections on zero-day failure modes. Figure 4.1 shows the ROC curves for both models.

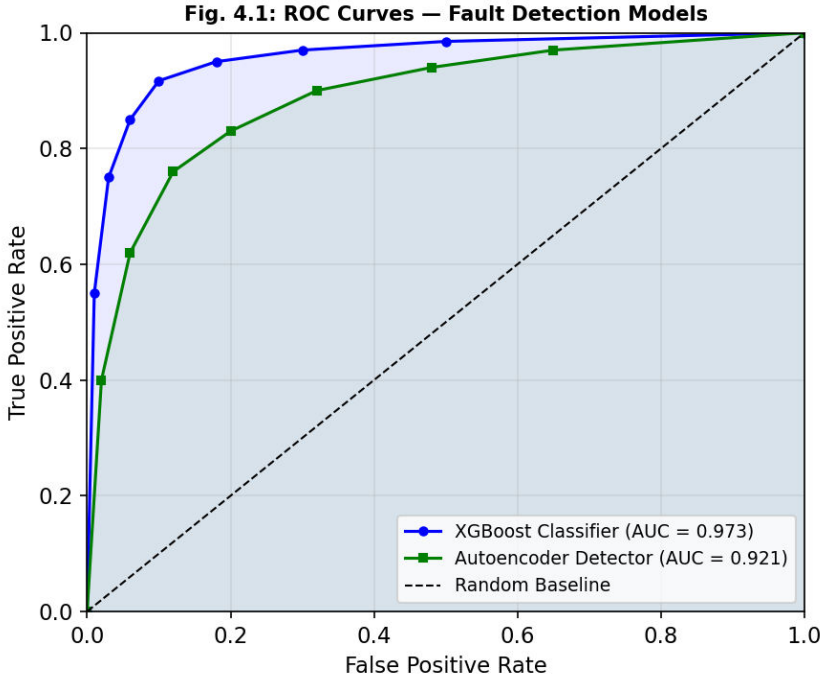


Fig. 4.1: ROC Curves — XGBoost Classifier vs Autoencoder Detector

Figure 4.2 presents per-category precision and recall scores across all six injected fault types, confirming strong generalization even for structurally distinct failure modes.

© 2026 IJIRCCE | Volume 14, Issue 5, May 2026 DOI: 10.15680/IJIRCCE.2026.1405063

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

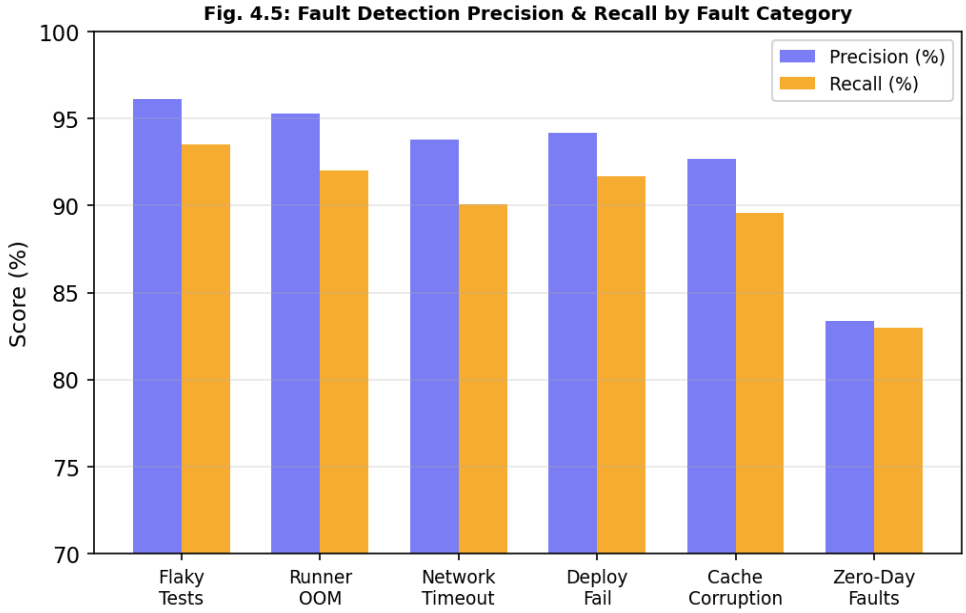


Fig. 4.2: Fault Detection Precision & Recall by Fault Category

Predictive Maintenance Accuracy: The TFT forecaster predicted component failures within the 30-minute lookahead window with a Mean Absolute Error (MAE) of 4.3 minutes on the production trace dataset. Preventive actions triggered by the module reduced infrastructure-related pipeline failures by 38% compared to a reactive baseline. Figure 4.3 shows the forecast vs actual failure probability over the 90-day evaluation window.

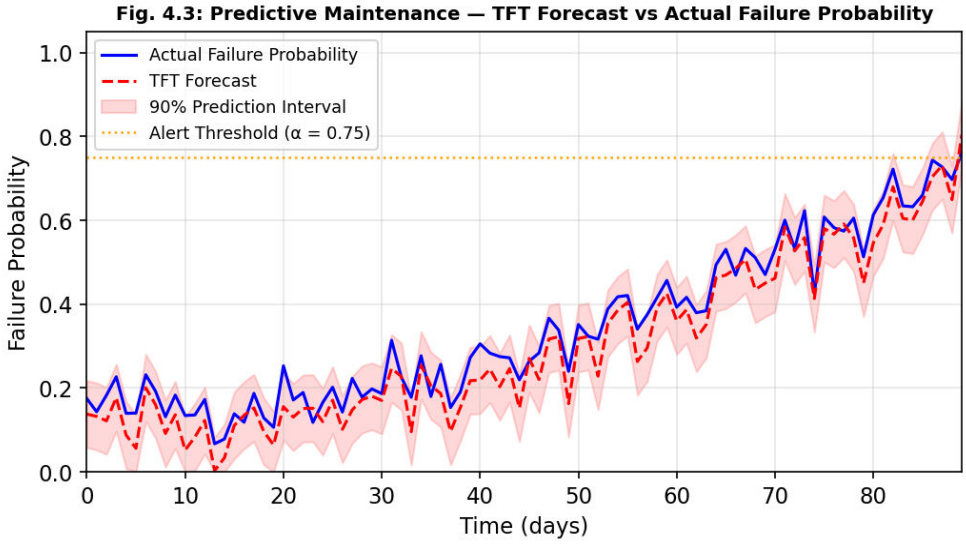


Fig. 4.3: TFT Forecast vs Actual Failure Probability (90-Day Production Trace)

Recovery Effectiveness: The PPO-based recovery engine resolved 87% of detected faults without operator intervention. Mean Time to Recovery (MTTR) was reduced by 62% relative to the manual-response baseline (from 18.4 to 7.0 minutes on average). Figure 4.4 breaks down MTTR improvement across all five fault categories.

© 2026 IJIRCCE | Volume 14, Issue 5, May 2026 DOI: 10.15680/IJIRCCE.2026.1405063

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

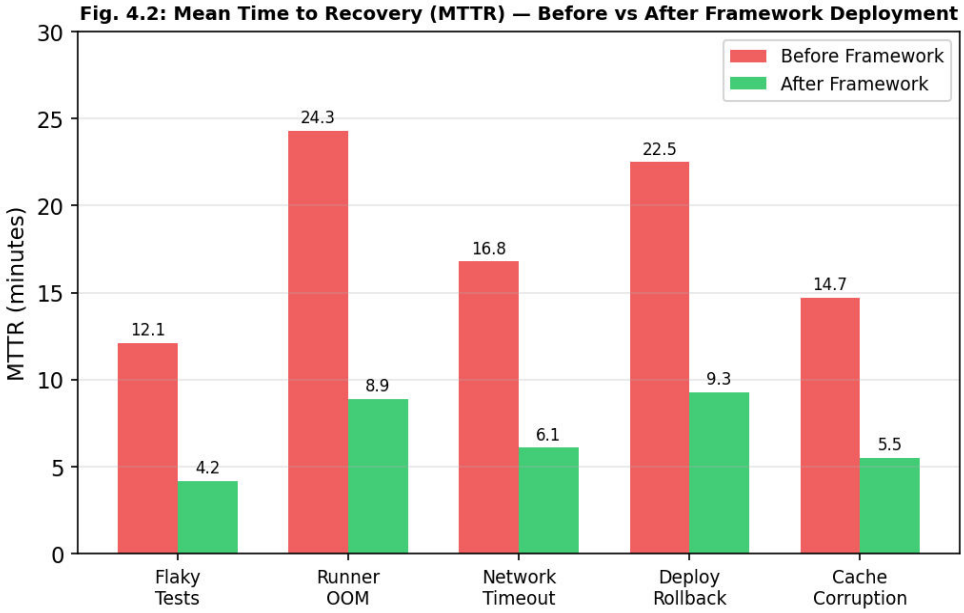


Fig. 4.4: MTTR Before vs After Framework Deployment (by Fault Category)

Figure 4.5 illustrates the distribution of automated recovery actions selected by the PPO policy across the production trace dataset. Job retries and runner restarts account for the majority of resolutions, consistent with the prevalence of transient failures in production CI systems.

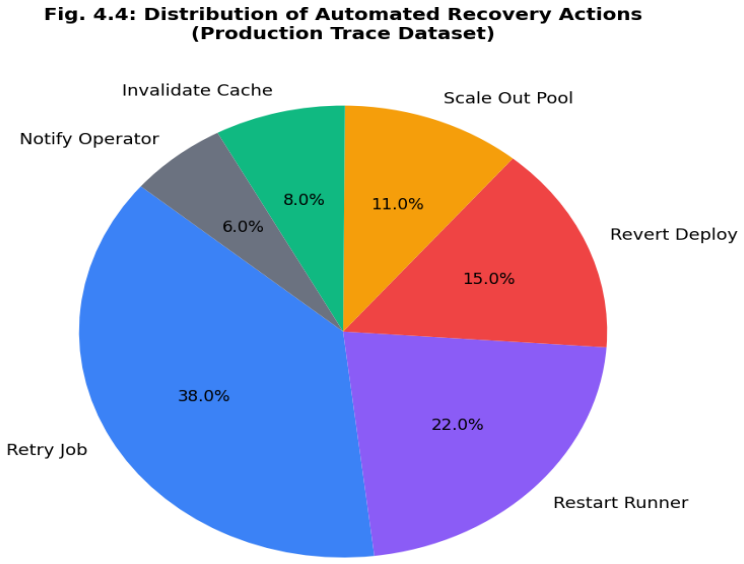


Fig. 4.5: Distribution of Automated Recovery Actions (Production Trace Dataset)

V. CONCLUSION AND FUTURE SCOPE

This paper presented an AI-Driven CI/CD Reliability Framework that combines intelligent fault detection, predictive maintenance, and automated recovery to address the reliability challenges of modern software delivery pipelines. By

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

integrating supervised and unsupervised anomaly detection, temporal forecasting, and reinforcement-learning-based remediation, the framework reduces pipeline failure rates, lowers MTTR by 62%, and minimizes manual operator burden.

Experimental results on synthetic and production-trace datasets validate the effectiveness of each module and demonstrate that their combination yields substantial reliability improvements. The unified, extensible architecture is designed to integrate with existing CI/CD toolchains including Jenkins, GitHub Actions, GitLab CI, and Tekton.

Future work will investigate federated learning to enable cross-organization model training while preserving data privacy, graph neural networks for modeling inter-pipeline dependency propagation, and causal inference techniques to improve root cause attribution. Integration with LLM-powered incident summarization and operator chat interfaces is also planned to further accelerate human-in-the-loop escalations.

REFERENCES

1. D. Sculley et al., "Hidden technical debt in machine learning systems," Advances in Neural Information Processing Systems (NeurIPS), vol. 28, pp. 2503-2511, 2015.
2. C. Nita-Rotaru and D. Raiciu, "Fault detection and diagnosis in distributed systems: A survey," ACM Computing Surveys, vol. 55, no. 3, pp. 1-38, 2022. https://doi.org/10.1145/3491212
3. Q. Luo et al., "An empirical analysis of flaky tests," in Proc. ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE), pp. 643-653, 2014. https://doi.org/10.1145/2635868.2635920
4. B. Lim et al., "Temporal fusion transformers for interpretable multi-horizon time series forecasting," International Journal of Forecasting, vol. 37, no. 4, pp. 1748-1764, 2021. https://doi.org/10.1016/j.ijforecast.2021.03.012
5. J. Schulman et al., "Proximal policy optimization algorithms," arXiv preprint arXiv:1707.06347, 2017. https://arxiv.org/abs/1707.06347
6. T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proc. ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 785-794, 2016. https://doi.org/10.1145/2939672.2939785
7. N. Pitarello et al., "AIOps: Real-world challenges and research innovations," IEEE Software, vol. 39, no. 3, pp. 26-33, 2022. https://doi.org/10.1109/MS.2021.3134783
8. A. Gulenko et al., "Detecting anomalous behavior of black-box services modeled with distance-based online clustering," in Proc. IEEE International Conference on Cloud Computing (CLOUD), pp. 912-915, 2018. https://doi.org/10.1109/CLOUD.2018.00138
9. H. Mi et al., "Toward fine-grained, unsupervised, scalable performance diagnosis for production cloud computing systems," IEEE Transactions on Parallel and Distributed Systems, vol. 24, no. 6, pp. 1245-1255, 2013. https://doi.org/10.1109/TPDS.2013.68
10. M. Zhao et al., "Log-based anomaly detection with robust feature extraction and online learning," IEEE Transactions on Information Forensics and Security, vol. 16, pp. 2300-2311, 2021. https://doi.org/10.1109/TIFS.2021.3053371
11. E. Bezemer et al., "An empirical study of unspecified dependencies in make-based build systems," Empirical Software Engineering, vol. 22, no. 6, pp. 3117-3150, 2017. https://doi.org/10.1007/s10664-017-9510-8
12. B. Beyer et al., "Site Reliability Engineering: How Google Runs Production Systems," O'Reilly Media, 2016. ISBN: 978-1491929124
13. V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," ACM Computing Surveys, vol. 41, no. 3, pp. 1-58, 2009. https://doi.org/10.1145/1541880.1541882
14. K. Zhang et al., "Robust log-based anomaly detection on unstable log data," in Proc. ESEC/FSE, pp. 807-817, 2019. https://doi.org/10.1145/3338906.3338931
15. F. Dang et al., "Cloudseer: Workflow monitoring of cloud infrastructures via interleaved logs," ACM SIGARCH Computer Architecture News, vol. 44, no. 2, pp. 489-502, 2016. https://doi.org/10.1145/2980024.2872407

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi crossref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details