

ARPE-IoT: Adaptive Recovery and Runtime Policy Enforcement for Resilient Cloud–Edge IoT Workflows

Swapneswar Sundar Ray

Independent Researcher

Randolph, NJ, USA

Email: swapneswar.ray@gmail.com

Abstract—Agentic artificial intelligence can improve Internet of Things (IoT) workflow resilience by using runtime context to select and adapt recovery actions across devices, edge gateways, cloud services, and enterprise applications. However, autonomously completed recovery actions may still violate authorization, privacy, data-residency, actuator-safety, cost, or operational constraints. This paper proposes ARPE-IoT, an Adaptive Recovery and Policy Enforcement Framework for Agentic IoT Workflows. ARPE-IoT treats agent-generated recovery actions as proposals that must pass hard runtime policies before execution. Permissible actions are ranked using soft operational utility, executed under a bounded recovery budget, validated against workflow and physical-state conditions, audited, and escalated to humans when autonomous resolution is unsafe. We implemented a lightweight Python simulator with temperature and occupancy sensors, two edge gateways, two cloud services, an HVAC actuator, a policy engine, and a constrained recovery planner. Across 10,800 simulated executions covering normal operation and eight injected failure classes, ARPE-IoT achieved a 100% safe handling rate, consisting of 82.2% autonomous resolution and 17.8% controlled safe-state escalation. It reduced policy violations from 22.2% for unconstrained agents and 33.3% for rule-based recovery to 0% in the evaluated scenarios, with 47.7 ms mean latency.

Index Terms—Agentic AI, Internet of Things, runtime policy enforcement, adaptive recovery, edge computing, workflow orchestration, governance.

I. INTRODUCTION

The Internet of Things increasingly supports operational workflows that span heterogeneous sensors, actuators, edge gateways, cloud services, analytical models, and enterprise applications. Unlike conventional centralized applications, these workflows execute across environments with different network conditions, computational capacities, security boundaries, and availability characteristics. A single workflow may depend on sensor observations, edge-based preprocessing, cloud inference, policy validation, and physical actuation. Consequently, device outages, communication failures, expired credentials, service degradation, anomalous data, and actuator errors can interrupt the workflow or place the physical process in an unsafe state.

Existing IoT platforms generally handle such failures through fixed retry limits, timeout rules, redundant components, checkpointing, rollback, or manually defined fallback paths [9]–[11]. These mechanisms remain valuable, but they

are typically configured before deployment and react to a limited set of anticipated conditions. Research on cloud and edge fault tolerance has similarly emphasized recovery, redundancy, resource scheduling, rerouting, and workload placement. However, the heterogeneity and dynamism of cloud–edge IoT environments make it difficult to encode an exhaustive recovery rule for every combination of device state, service dependency, network condition, and operational objective.

Agentic artificial intelligence introduces a possible alternative. An AI agent can observe runtime context, reason over workflow dependencies, generate recovery candidates, and coordinate actions across devices and services. For example, an agent may replace an unavailable sensor, redirect computation to another edge gateway, select an alternate cloud service, refresh a credential, roll back a command, or temporarily operate the workflow in a degraded mode. Such context-sensitive recovery can be more flexible than static retry logic, particularly when the failure was not explicitly anticipated during system design.

Nevertheless, introducing an autonomous agent into an IoT control loop also creates substantial risk. A recovery action that restores technical availability may still violate authorization, privacy, data-residency, safety, cost, or operational policies. An agent might route sensitive sensor data to an unapproved cloud region, invoke an unauthorized device, retry a physical command beyond a safe limit, or select an alternative actuator whose operating range is incompatible with the current environment. Because IoT workflows may influence physical systems, recovery cannot be judged solely by whether execution resumes.

Recent agent-governance research has therefore emphasized enforceable runtime constraints rather than relying exclusively on prompts, model alignment, or post-execution audits [1]–[3]. Proposed approaches validate actions at policy gates, monitor execution traces, restrict tool permissions, and route high-risk decisions to human operators. These studies establish that agent-generated actions should be treated as proposals until they have passed deterministic or machine-verifiable checks. However, most emerging agent-governance architectures target general tool-using agents, enterprise workflows, cybersecurity operations, or cloud data pipelines rather than failure recovery in heterogeneous IoT workflows.

This paper addresses the resulting gap by proposing ARPE-IoT, an Adaptive Recovery and Policy Enforcement Framework for Agentic IoT Workflows. When a failure occurs, ARPE-IoT classifies the failure, generates candidate recovery actions, rejects actions that violate hard constraints, ranks the remaining candidates according to recovery utility, and executes the highest-ranked permissible action. Recovery is bounded by retry limits, risk thresholds, validation requirements, and escalation conditions to prevent uncontrolled autonomous loops.

The framework distinguishes between hard policies and soft operational objectives. Hard policies express constraints that cannot be traded for improved availability, including authorization boundaries, actuator safety limits, prohibited data destinations, and mandatory approval requirements. Actions violating these constraints are rejected before execution. Soft objectives—including latency, expected recovery probability, cost, and energy usage—are used only to rank actions that have already passed hard-policy validation. This separation avoids treating safety or compliance as merely another weighted factor that can be outweighed by performance.

Unlike conventional IoT recovery systems that select actions using static exception rules, and unlike unconstrained agentic systems that execute model-generated plans directly, ARPE-IoT separates adaptive recovery planning from deterministic runtime authorization. The framework combines policy filtering before ranking, explicit allow/deny/escalate outcomes, bounded replanning, validation after each action, and safe-state escalation when no admissible action exists. The main contributions are: (i) a cloud–edge architecture for policy-constrained adaptive recovery in agentic IoT workflows; (ii) a recovery-selection method that first applies hard-policy filtering and then ranks permissible actions using operational utility; (iii) a bounded recovery protocol incorporating validation, safe-state transition, audit logging, and escalation; and (iv) a reproducible simulation comparing static retry, rule-based recovery, unconstrained agentic recovery, and ARPE-IoT under injected IoT failures.

II. RELATED WORK

A. Fault Tolerance in IoT and Cloud–Edge Systems

Fault tolerance in distributed IoT systems commonly uses redundancy, replication, retry, checkpointing, rollback, task rescheduling, and migration [12], [13]. Fog and edge architectures can detect failures close to the data source and reduce dependence on distant cloud infrastructure. Machine-learning-based fault detection has also been used to identify anomalous or faulty sensor data and initiate recovery at the fog layer.

Cloud workflow research has proposed fault-aware schedulers that map or remap tasks according to resource reliability, data locality, execution cost, and workload constraints [9], [10], [14]. These techniques improve workflow completion and infrastructure utilization, but the recovery alternatives are generally chosen by deterministic schedulers or predefined optimization procedures. They do not ordinarily reason over

dynamically expressed organizational policies or generate new recovery plans from heterogeneous runtime context.

IoT–edge–cloud environments introduce further challenges because devices may be resource constrained, intermittently connected, mobile, or managed by different administrative domains [15], [16]. Recent studies identify workload placement, communication reliability, latency, energy consumption, security, and fault recovery as interconnected orchestration concerns. Adaptive caching, delay-tolerant communication, and edge–cloud collaboration can improve availability, although these mechanisms do not by themselves prevent an autonomous recovery decision from violating a safety or governance requirement.

B. Self-Adaptive and Agentic Recovery

Self-adaptive systems observe their operating environment and modify their behavior in response to runtime changes. In IoT and fog systems, adaptive mechanisms have been applied to scheduling, resource allocation, anomaly response, load balancing, and service placement. Such systems generally operate over a predefined adaptation space, with policies or utility functions determining which configuration should be selected.

Agentic AI broadens this adaptation space by allowing a reasoning component to interpret operational context, construct multi-step plans, invoke tools, and coordinate specialized agents [5]–[8]. Emerging work on agentic cloud operations demonstrates that bounded agents can analyze telemetry and propose recovery actions such as resource reconfiguration, schema reconciliation, and failure remediation. One recent cloud-pipeline architecture reported lower recovery time and reduced manual intervention by validating agent actions against declarative governance policies. Its focus, however, is data-pipeline operation rather than sensor-to-actuator IoT workflows and cyber-physical recovery.

C. Runtime Policy Enforcement for Autonomous Agents

Policy-based control has a long history in multi-agent systems [1], [4]. Recent agentic architectures apply governance at tool boundaries, pre-action gates, runtime monitors, and post-action audit points. Authenticated-workflow approaches protect prompt, tool, data, and context boundaries of agentic systems and reject operations that lack required authorization or integrity evidence. Governance-by-architecture frameworks represent constraints as executable specification objects and place enforcement points directly within the agent loop. This is important because finite reward penalties or prompt instructions cannot guarantee that hard requirements will be respected.

Trace-oriented frameworks have also been developed to check an agent’s sequence of tool invocations against temporal, argument-level, and information-flow constraints. Such work demonstrates that some violations can only be identified by examining relationships across multiple actions rather than validating each tool call independently. These approaches provide useful policy and monitoring foundations, but they

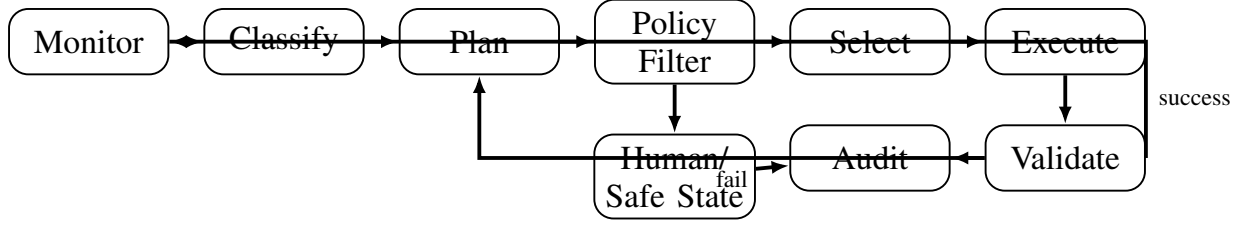


Fig. 1. ARPE-IoT execution order: policy filtering precedes selection; validation failure replans, success returns to monitoring, and mandatory escalation enters a safe state.

TABLE I
RESEARCH-GAP COMPARISON

Approach	Adapt.	Hard Gate	IoT	Post-Val.	Bound Esc.	
Static retry	No	Lim.	Lim.	No	Retry	Adapt.: adaptive;
Rule-based IoT	Lim.	Dep.	Yes	Some	Config.	
Cloud scheduling	Opt.	Lim.	Part.	Res.	Part.	
General agents	Yes	Prompt	No	Lim.	Var.	
Agent governance	Part.	Yes	No	Yes	Often	
ARPE-IoT	Yes	Yes	Yes	Yes	Yes	

Hard Gate: hard policy gate; Post-Val.: post-validation; Bound Esc.: bounded escalation; Lim.: limited; Dep.: implementation-dependent; Opt.: optimization-based; Part.: partial; Res.: resource-focused; Var.: variable.

do not fully address how a constrained agent should recover a failed IoT workflow.

Unlike general runtime-governance approaches such as AgentSpec [1] and Pro2Guard [2], ARPE-IoT embeds hard policy enforcement within a bounded IoT recovery loop that also includes post-action physical-state validation and safe-state escalation.

D. Research Gap

The literature contains mature mechanisms for distributed fault tolerance and emerging mechanisms for agent governance, but the two areas remain insufficiently integrated. Conventional IoT recovery is generally reliable but inflexible, while unconstrained agentic recovery can be adaptive but operationally unsafe. Existing governance frameworks can approve or reject agent actions, yet they usually do not define an end-to-end recovery protocol for heterogeneous cloud-edge IoT workflows. To the best of our knowledge, existing work has not jointly addressed context-aware agentic recovery, hard runtime policy enforcement, bounded re-planning, degraded-mode execution, and post-action state validation within a unified cloud-edge IoT workflow architecture.

III. SYSTEM MODEL AND PROBLEM FORMULATION

An IoT workflow is modeled as $W = (V, E_w, \sigma)$, where V is the set of workflow tasks, $E_w \subseteq V \times V$ is the dependency relation, and $\sigma : V \rightarrow \{D, G, C, M, A\}$ maps tasks to devices, gateways, cloud services, models, or actuators. Runtime state is s_t , and a failure event is $f = (\tau, v, \rho, t)$, where τ is the failure type, v is the affected task, ρ is severity, and t is time. Failure types include sensor outage, gateway timeout, cloud timeout, expired token, anomalous data, data-residency violation, excessive retry, and unsafe actuator command.

The prototype uses a constrained planning agent that maps structured failure context to candidates from a fixed action catalogue

$$A_R = \{a_{retry}, a_{sub}, a_{route}, a_{roll}, a_{deg}, a_{esc}\}. \quad (1)$$

In the current prototype, the planning component is a constrained context-aware planner rather than an unconstrained generative LLM. Agentic behavior refers to selecting, ranking, and replanning recovery actions from the fixed catalogue using runtime failure context. The constrained planner receives a structured failure vector and generates a ranked subset from the catalogue using the same utility estimator used in evaluation. The unconstrained-agent baseline uses the same candidate generator and utility estimator but bypasses hard-policy filtering, post-action validation, and safe-state enforcement. The planner may recommend actions, but cannot directly invoke actuators or cloud services. A policy p_i returns $\{allow, deny, escalate\}$ over action, state, identity, data attributes, and physical context. Conflicts are resolved by $deny \succ escalate \succ allow$. Hard-policy feasibility and human-review sets are

$$A_f = \{a \in A_C \mid \forall p_i \in P_H, p_i(a, s_t, x) = allow\}, \quad (2)$$

$$A_h = \{a \in A_C \mid \exists p_i \in P_H, p_i(a, s_t, x) = escalate\}. \quad (3)$$

A. Recovery Utility Estimation

Feasible actions are ranked only by operational utility:

$$U(a) = w_s \hat{S}(a) - w_l L(a) - w_c C(a) - w_r R(a) - w_e E(a), \quad (4)$$

where all terms are normalized to $[0, 1]$, $\sum_j w_j = 1$, and $w_j \geq 0$. Experiments use $w_s = .40$, $w_l = .20$, $w_c = .10$, $w_r = .25$, and $w_e = .05$. The success estimate uses smoothed historical outcomes, $\hat{S}(a, f) = (n_{succ}(a, f) + \alpha) / (n_{att}(a, f) + \alpha + \beta)$, with $\alpha = \beta = 1$. Estimates are calibrated on a separate warm-up simulation, frozen before evaluation, and shared by ARPE-IoT and the unconstrained-agent baseline to avoid order effects. The recovery budget is $B = (N_{max}, T_{max})$, where N_{max} bounds attempts and T_{max} bounds total recovery duration. If $A_f = \emptyset$, a mandatory-review policy applies, a timeout occurs, or B is exhausted, the workflow enters S_{safe} : no new actuator command is issued, the last safe set point is retained, restricted data flows are stopped, and an operator is alerted. Actions in A_h are excluded from autonomous selection and require operator approval.

Algorithm 1 ARPE-IoT Bounded Recovery Protocol

Require: Failure event f , state s_t , candidate catalogue A_R , policies P_H , budget $B = (N_{max}, T_{max})$

Ensure: Recovered workflow state or safe-state escalation

- 1: Initialize attempted-action history $H \leftarrow \emptyset$, attempt count $N_{attempts} \leftarrow 0$, start time t_0 , and elapsed time $T \leftarrow 0$.
- 2: Classify $f = (\tau, v, \rho, t)$ and update workflow state s_t .
- 3: **while** $N_{attempts} < N_{max}$ and $T < T_{max}$ **do**
- 4: Generate candidate set $A_C \subseteq A_R$ from structured failure context within planner timeout T_p .
- 5: Compute feasible set A_f and human-review set A_h using hard policies.
- 6: **if** $A_f = \emptyset$ or mandatory escalation condition holds **then**
- 7: Enter S_{safe} , write audit record, and notify operator.
- 8: **return**
- 9: **end if**
- 10: **if** $A_f \setminus H = \emptyset$ **then**
- 11: Enter S_{safe} , write audit record, and notify operator.
- 12: **return**
- 13: **end if**
- 14: Select $a^* \leftarrow \arg \max_{a \in A_f \setminus H} U(a)$.
- 15: Execute a^* and update state s_{t+1} .
- 16: **if** workflow, physical-state, and policy-log validation succeeds **then**
- 17: Resume workflow and write audit record.
- 18: **return**
- 19: **end if**
- 20: Set $H \leftarrow H \cup \{a^*\}$, $N_{attempts} \leftarrow N_{attempts} + 1$, and $T \leftarrow$ current time $- t_0$.
- 21: **end while**
- 22: Enter S_{safe} , stop unsafe actions, and escalate to a human operator.

Trusted components are the policy store, enforcement engine, audit logger, identity provider, and actuator safety controller. Sensors, networks, cloud APIs, credentials, edge nodes, and planning agents may fail or be compromised. Policy integrity is assumed for this study.

IV. ARPE-IoT FRAMEWORK

Fig. 1 shows the framework. A monitor collects telemetry, events, policy outcomes, and workflow state. A failure-classification agent estimates type and severity. A planning agent proposes candidate actions, but a policy engine filters hard-policy violations before the selector ranks remaining actions. An execution and validation agent applies the action, checks state consistency, downstream safety, and policy compliance, and either resumes the workflow, replans within budget, or escalates.

V. PROTOTYPE AND EXPERIMENTAL METHODOLOGY

We implemented a discrete-event Python simulator containing one temperature sensor, one occupancy sensor, two edge gateways, two cloud services, one HVAC actuator, one policy engine, and one constrained recovery planner. The simulator, policy definitions, failure configurations, and result-generation script are provided in the supplementary artifact files `simulate_arpe_iot.py`, `results.json`, and `README.md`. Experiments used Python 3, random seed 42, $N_{max} = 3$, $T_{max} = 200$ ms, planner timeout $T_p = 50$ ms, and the weights above. The run count is $(1 + 8) \times 300 \times 4 = 10,800$: one normal case plus eight failure classes, 300 repetitions, and four methods.

Baselines are static retry, conventional rule-based recovery, unconstrained agentic recovery, and ARPE-IoT. The rule-based baseline contains technical recovery rules but no independent runtime policy gate; therefore, it may select

TABLE II
AGGREGATE RESULTS OVER 10,800 SIMULATED RUNS

Method	Safe %	Auto. Resol. %	Esc. %	Viol. %	Lat. ms	False %	
Static	64.6	64.6	0.0	11.1	63.4	7.6	Auto. Resol.:
Rule	58.5	58.5	0.0	33.3	36.0	30.0	
Agent	69.8	58.7	11.1	22.2	36.6	20.2	
ARPE	100.0	82.2	17.8	0.0	47.7	0.0	

autonomous resolution; Esc.: escalation; Viol.: policy violations; Lat.: mean latency; False: false recovery.

TABLE III
SAFE HANDLING BY FAILURE CLASS (%)

Failure	Static	Rule	Agent	ARPE
Sensor outage	64.0	90.3	88.3	100.0
Gateway timeout	65.7	85.3	88.3	100.0
Cloud timeout	71.3	83.7	84.7	100.0
Expired token	69.7	93.0	91.7	100.0
Data anomaly	66.7	76.0	75.7	100.0
Residency violation	72.7	0.0	0.0	100.0
Excessive retry	0.0	0.0	100.0	100.0
Unsafe command	73.0	0.0	0.0	100.0

predefined backup services or actuator actions that become noncompliant under the current context. Metrics are Safe Handling Rate $= (N_{safe_autonomous_resolution} + N_{controlled_safe_state_escalation}) / N_{total}$, Autonomous Resolution Rate $= N_{resolved_without_human_intervention} / N_{total}$, policy-violation rate, mean latency, escalation, and false recovery, where false recovery denotes an apparently successful but unsafe or noncompliant recovery. For baselines, autonomous resolution may include unsafe or noncompliant technical completions; for ARPE-IoT, it is also safe recovery because policy filtering and validation are active.

These comparison methods are controlled variants of the same simulation environment rather than independent external implementations. The unconstrained-agent configuration serves as a direct ablation of policy filtering, post-action validation, and safe-state enforcement, while the rule-based configuration provides a controlled non-agentic baseline within the same simulation environment.

VI. RESULTS AND DISCUSSION

Table II shows that ARPE-IoT achieved a 100% safe handling rate by resolving 82.2% of cases autonomously and routing 17.8% to controlled safe-state escalation. This does not mean every workflow completed its intended business operation autonomously. Instead, all evaluated workflows either recovered without violating policy or stopped in S_{safe} . The observed 0% executed policy-violation rate is an expected consequence of the hard pre-execution policy gate under the trusted-policy-engine assumption; the experiment verifies that this invariant was maintained across the evaluated scenarios rather than treating zero violations as an unexpected statistical result. Compared with rule-based recovery, ARPE-IoT improved safe handling by 41.5 percentage points and reduced policy violations from 33.3% to 0% in the evaluated scenarios. Compared with unconstrained agentic recovery, it improved

TABLE IV
SENSITIVITY AND ABLATION SUMMARY

Configuration	Safe (%)	Esc. (%)	Viol. (%)
$w_r = .10, w_s = .55$	100.0	17.5	0.0
$w_r = .25, w_s = .40$	100.0	17.8	0.0
$w_r = .40, w_s = .25$	100.0	18.1	0.0
No policy filter	84.7	18.0	15.3
No safe-state escalation	88.8	11.2	0.0

safe handling by 30.2 percentage points and eliminated policy-violating data-routing and actuator actions in the evaluated scenarios. Across reported non-boundary proportions, the maximum 95% confidence-interval half-width was 1.9 percentage points; boundary values such as 0% are interpreted only within the evaluated scenarios.

Table III shows that aggregate gains are concentrated in policy-sensitive failures. Rule-based recovery handled ordinary sensor and token failures well, but failed on residency violations and unsafe commands because its embedded fallback rules lacked a separate policy gate. ARPE-IoT preserved safety in those cases by denying noncompliant actions and escalating when no admissible autonomous action remained. Static retry was counted as safe in unsafe-command scenarios only when retry exhaustion prevented command execution or the simulated context returned to an admissible state. The unconstrained agent handled excessive-retry cases by selecting a non-retry alternative from the shared catalogue; unlike ARPE-IoT, this behavior was not guaranteed by an enforceable retry policy. ARPE-IoT latency averaged 47.7 ms (SD 46.9 ms, median 22.5 ms, 95th percentile 144.5 ms), compared with 36.6 ms for the unconstrained agent. The overhead reflects policy evaluation, validation, and escalation. The difference between mean and median latency reflects a right-skewed distribution caused by repeated recovery attempts and escalation in a minority of cases.

Sensitivity and ablation results are summarized compactly in Table IV. In sensitivity runs, w_r was varied while w_s was adjusted by the same amount so weights continued to sum to one. Increasing w_r from .10 to .40 preserved safe handling while changing escalation only slightly on the shared test traces. Ablations show that removing the policy filter reintroduced 15.3% policy violations and reduced safe handling to 84.7%, while removing safe-state escalation reduced safe handling to 88.8%. Limitations include the simulation-to-reality gap, candidate-catalogue completeness, non-adversarial sensors, and the assumption that the policy engine is uncompromised.

VII. CONCLUSION

This paper presented ARPE-IoT, a framework for adaptive recovery and runtime policy enforcement in agentic IoT workflows. In 10,800 simulated executions, ARPE-IoT improved safe handling over static retry, rule-based recovery, and unconstrained agentic recovery while reducing policy violations to 0% in the evaluated scenarios; 17.8% of these safe outcomes

were controlled escalations rather than autonomous business-task completion. The results support separating hard policy feasibility from soft recovery utility. Future work will implement ARPE-IoT in a physical testbed, calibrate utility estimates from operational data, formally verify policy predicates, and evaluate human-in-the-loop escalation quality.

REFERENCES

- [1] H. Wang, C. M. Poskitt, J. Sun, Y. Liu, and J. S. Dong, "AgentSpec: Customizable runtime enforcement for safe and reliable LLM agents," arXiv:2503.18666, 2025.
- [2] Z. Gong, Y. Peng, J. Zhang, L. Li, X. Zhang, and L. Chen, "Pro2Guard: Proactive runtime enforcement for LLM agents via probabilistic model checking," arXiv:2505.12849, 2025.
- [3] N. Karande, S. Desai, and P. Shah, "Authenticated delegation and authorized AI agents," arXiv:2501.09674, 2025.
- [4] A. Bussola, I. Asadi, and G. Spanakis, "Governance by architecture: Structuring AI multi-agent systems for responsibility and control," arXiv:2507.15565, 2025.
- [5] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafraan, K. Narasimhan, and Y. Cao, "ReAct: Synergizing reasoning and acting in language models," in *Proc. ICLR*, 2023.
- [6] T. Schick et al., "Toolformer: Language models can teach themselves to use tools," in *Proc. NeurIPS*, 2023.
- [7] M. Radhamani, E. Shenoy, and Y. Srivastava, "The rise of agentic AI: A review of frameworks, protocols and applications," *Future Internet*, vol. 17, no. 9, 2025.
- [8] S. K. L. Wang, M. P. S. Bhatia, and M. B. Blake, "A survey of context engineering for large language models," arXiv:2507.13334, 2025.
- [9] A. Taghinezhad-Niar and J. Taheri, "Fault-tolerant cost-efficient scheduling for energy and deadline-constrained IoT workflows in edge-cloud continuum," *IEEE Trans. Services Comput.*, vol. 18, no. 5, pp. 2892–2903, 2025, doi: 10.1109/TSC.2025.3599497.
- [10] K. M. Uma and S. Shukla, "Energy and performance-aware workflow scheduler using dynamic virtual network resource optimization under edge-cloud platform," *Comput. Electr. Eng.*, vol. 123, pt. B, Art. no. 110085, 2025, doi: 10.1016/j.compeleceng.2025.110085.
- [11] I. Afzal, N. ul Amin, Z. Ahmad, and A. Algarni, "A latency-aware and fault-tolerant framework for resource scheduling and data management in fog-enabled smart city transportation systems," *Comput., Mater. Continua*, vol. 82, no. 1, pp. 1377–1399, 2025, doi: 10.32604/cmc.2024.057755.
- [12] Z. Xiong, S. Meng, S. Yao, and L. Luo, "Fault tolerant and intelligent reliability assurance in cloud-edge systems," in *Proc. 12th Int. Conf. Dependable Syst. Appl. (DSA)*, 2025, pp. 156–162, doi: 10.1109/DSA66321.2025.00028.
- [13] Praveen K. S., L. Manjunatha Rao, and Rajshekhar Ghogge, "Cost-reliability-aware multi-workflow scheduling using multi-agent deep reinforcement learning in edge-cloud systems," *Ingénierie des Systèmes d'Information*, vol. 31, no. 3, pp. 889–899, 2026, doi: 10.18280/isi.310320.
- [14] L. R. Raju, M. V. K. Reddy, S. R. Surukanti, G. Sudhakar, V. V. Subrahmanya Sarma M, and A. Adepu, "IntelliScheduler: An edge-cloud computing environment hybrid deep learning framework for task scheduling based on learning," *Sci. Rep.*, vol. 16, Art. no. 11219, 2026, doi: 10.1038/s41598-026-41330-8.
- [15] D. Kapoor, D. Gupta, and M. Uppal, "Analyzing the impact of edge, fog and cloud computing on predictive maintenance in the Industrial Internet of Things," *Discover Computing*, vol. 28, Art. no. 207, 2025, doi: 10.1007/s10791-025-09653-8.
- [16] K. Swathi, P. Durga, K. V. Prasad, A. K. Chaitanya, K. Santhi, P. Vidyullatha, and S. V. A. Rao, "Secure blockchain integrated deep learning framework for federated risk-adaptive and privacy-preserving IoT edge intelligence sets," *Sci. Rep.*, vol. 15, Art. no. 41133, 2025, doi: 10.1038/s41598-025-24895-8.