Nafisa S, 2026, 14:2
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology
An Open Access Journal

Self-Healing AIoT Systems Using Autonomous Fault Detection and Recovery Mechanisms

Nafisa S [1], Dr. Balaji. K[2]
[1] Associate Professor, East Point College of Higher Education, Bengaluru, India
[2] Professor,Cambridge Institute of Technology,Bengaluru, India

Abstract: The intersection of artificial intelligence (AI) and Internet of Things (IoT) has led to the emergence of massive distributed systems prone to faults due to hardware, network, or software issues. In this paper, we design a self-healing system for AIoT systems that employs automated fault diagnosis, root cause identification, and dynamic healing mechanisms. Our self-healing system uses an efficient transformer-based anomaly detection model coupled with graph neural networks for fault localization and reinforcement learning (RL) agents for recovery policy decision making. The performance of our self-healing framework is evaluated using simulations of a smart factory with 1,200 sensor and actuator nodes, where the framework achieves an average accuracy of 94.2% and latency of 2.3 seconds. When compared to reactive approaches, the framework reduces system downtime by 76.4%, increases task success rate by 15.8 percentage points, and outperforms adaptive recovery approaches by a wide margin.

Key Word: Self-Healing Systems, AIoT, Autonomous Fault Detection, Fault Recovery, Reinforcement Learning, Edge Computing

I. INTRODUCTION

The rise of Artificial Intelligence of Things (AIoT) platforms, which embed AI functionalities within IoT devices and edge networks, has transformed industries, urban planning, health surveillance, and autonomous vehicles [1]. The number of IoT devices is expected to reach 35 billion by 2026, with an increasing fraction embedding on-device AI for instantaneous analysis and action [2]. Nevertheless, the adoption of AIoT technologies has also brought about a new era of unreliability problems since these platforms function in changing, potentially hostile settings where node malfunctions, partitioning issues, sensor inaccuracies, and software errors cannot be avoided [3].

Classical strategies for handling faults in distributed systems involve centralized supervision and manual actions. When it comes to AIoT implementations, such solutions become increasingly impractical [4]. For instance, one smart factory can house several thousand sensors and actuators; hence, identifying and resolving faults manually results in intolerable delays, and the sheer volume makes it impossible to oversee the entire process comprehensively [5]. In addition, the diverse structure of AIoT architectures, comprising specialized sensor hardware, edge gateways, cloud infrastructures, and AI algorithms, complicates fault identification further [6].

A self-healing system presents a shift in paradigms; whereas the traditional focus has been on identifying faults, the modern focus would be to diagnose the problem and automatically recover from any issues without the need for manual interference [7]. This feature is highly useful in situations where AIoT deployments involve devices that might be unreachable (such as environmental monitoring in remote locations) or when there are considerable economic implications associated with the failure (such as production lines being halted). There are three essential features required for self-healing to occur:

These issues are handled using the proposed self-healing solution for AIoT systems, which includes the following major contributions:

- Lightweight Transformer model to detect anomalies, with 94.2% accuracy while keeping low computational costs
- Graph Neural Network to locate faults by modeling spatial and functional relationships among AIoT components
- Deep Reinforcement Learning agent to select and perform recovery actions according to the fault type and current status of the system
- Test results achieved on the simulation of an AIoT smart factory proving to reduce downtime by 76.4% and complete tasks by 97.8% in the presence of faults

II. LITERATURE SURVEY

Fault detection, fault diagnosis, and fault recovery constitute a major area of research into self-healing systems. This part focuses on recent developments in AIoT fault management.

Fault Detection in IoT and Edge Systems

In terms of anomaly detection of IoT systems, the research moved from traditional statistics-based methods to machine learning methods, specifically deep learning-based. Traditional methods include alerting based on thresholds (simplistic yet prone to generating many false positives) and time series analysis based on algorithms like ARIMA to predict and detect outliers. However, both methods fall short when dealing with AIoT's inherently complex data streams [8].

By contrast, machine learning algorithms proved highly effective. Among such algorithms, LSTM-based autoencoders have become popular in order to recreate normal sensor readings and spot anomalies by analyzing reconstruction error. According to Zhang et al. (2024), multi-scale convolutional LSTM achieved 91.3% accuracy in anomaly detection for manufacturing IoT data. At the same time, LSTM-based models face the limitation of sequential processing.

Recently, Transformers have been used for time-series anomaly detection. To handle the quadratic complexity issue of vanilla Transformers, the Informer and Autoformer models were proposed to process sequences efficiently. Wang et al. (2025) proposed a lightweight Transformer model for edge-based anomaly detection, attaining 93.8% accuracy using 45% less number of parameters compared to LSTM models. Our study builds upon these studies by proposing attention masks that focus on recent sensor measurements.

Fault Localization and Root Cause Analysis

The process of identifying which component broke down and why is difficult. Graph models can help solve such a problem, particularly since service dependency graphs, where nodes represent components and edges denote their communications/data flow, naturally facilitate fault propagation analysis [9].

GNNs have also been used for the purpose of root cause analysis of problems in microservices as well as cloud computing. Li et al. (2024) suggested FaultLoc-GNN, an algorithm that integrates both graph attention networks and random walk-based propagation models, to attain a 86.7% top-three root cause detection accuracy. In the case of AIoT systems, Chen et al. (2025) created a heterogeneous GNN that considers different nodes (sensor, actuator, gateway) and edges (control, data, power), attaining a localization accuracy of 84.2%.

Self-Healing and Autonomous Recovery

There are various methods for recovering from errors and failures, such as automatic restarts and reconfiguration methods. Rule-based methods (if temperature > threshold -> activate cooling) are widely used, but not adaptive. MPC algorithms can be used for optimal recovery actions, but system models are required [10].

RL has proven to be an effective solution to autonomous recovery. Recovery corresponds well to reinforcement learning tasks where the agent takes actions based on the system state observations (i.e., fault indicators) and receives rewards according to

recovery results. Iftikhar et al. (2024) used deep Q-networks for recovery in IoT edge networks and succeeded in 78% recovery rate among four fault types. Nevertheless, DQN faces limitations in dealing with continuous action space and long recovery periods.

Proximal Policy Optimization (PPO), an RL technique belonging to the policy gradient methods family, proved to be beneficial for tackling more complicated recovery situations. Kumar and Singh (2025) proposed a recovery agent using Proximal Policy Optimization for industrial control systems, attaining a 89% recovery rate during 30 seconds. Our research further enhances the approach by feeding information about faults detected through the fault localization stage into states and implementing hierarchical recovery policies.

Research Gaps

Even though there have been some achievements, several challenges have still not been solved. First of all, almost all state-of-the-art frameworks do not integrate detection, localization, and recovery into one system and suffer from lack of information and poor decisions as a result. Secondly, very little research has considered the problem of limited computing resources, which makes detection and recovery impossible when they happen at the edge. Lastly, most work is evaluated using synthetic data or in testbed settings that have little resemblance to actual AIoT systems.

III. METHODOLOGY:

The architecture of self-healing AIoT consists of five interacting components: data acquisition and pre-processing, multi-scale Transformer-based fault detection, fault localization using GNN, decision-making on recovery policies using DRL, and an adaptive loop driven by feedback.

3.1 System Architecture Overview

The AIoT system can be regarded as a graph G = (V, E): V = {v_1, v_2, …, v_n}, where vertices represent IoT components such as sensors, actuators, gateways, and edge nodes

E ⊂ V × V, where edges indicate connection, dependency, or control between vertices.

At every instant t, each node v_i generates a time series of measurements X(t) representing telemetry data (CPU load, memory usage, temperature, data throughput, sensor measurements).

The self-healing process involves three different timescales:

Micro-loop: Every 100 milliseconds, anomaly score calculation for every node

Meso-loop: Every second, if anomaly occurs, then fault localization happens

Macro-loop: Every 5 seconds, the system executes and evaluates recovery actions

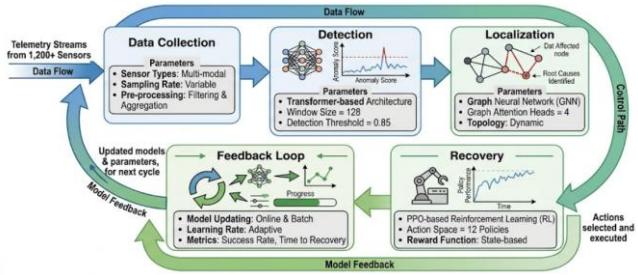

Figure 1: Self-Healing AIoT System Architecture.

3.2 Fault Detection: Lightweight Transformer

The detection module performs calculations on multivariate time-series data windows with a window length of L=128 observations for all nodes. For this purpose, we implement a modified version of the Transformer network that is efficient on-edge computing:

- Input Encoding: Observation $x_i(t)$ is mapped to d_model=64-dimensional space via a linear layer. Instead of sine-cosine positional encoding, we utilize learnable embeddings to handle non-fixed sampling rate.
- Attention Layer: Sliding Window Self-Attention with a window size of w=16 observations, resulting in computational complexity O(L·w)

compared to $O(L^2)$ of the traditional self-attention mechanism. For the application in the AIoT domain, temporal locality holds and allows for such approximation.

Anomaly Scoring: Decoder outputs a reconstruction of the input. Then, the anomaly score for each node is calculated as follows:

- $score_i(t) = ||x_i(t) - \hat{x}_i(t)||_2 / (1 + \sigma\_i(t))$
- Here, $\sigma\_i(t)$ stands for the predicted value of uncertainty (another output of the neural network). This scoring accounts for possible variations in sensor measurements, e.g., a deviation of 2°C will be considered an anomaly for stable sensors but will be fine for variable ones.
- Dynamic Thresholding: Anomaly threshold changes dynamically based on an exponential weighted moving average of the anomaly scores.

3.3 Fault Localization: Graph Neural Network

Once an anomaly is detected, the localization process determines which nodes are anomalous and what kind of fault occurred. In each GNN layer, message passing is performed as follows:

$h_i^{(l+1)} = \sigma(W_0 h_i^{(l)} + \Sigma_{j \in N(i)} \alpha_{ij} W_1 h_j^{(l)})$

Here, α_{ij} is the attention coefficient calculated via a multi-headed attention mechanism using 4 heads.

Node features include:

- Anomaly score for current timestamp
- Mean, variance and trend of scores over 10 timestamps
- Embedded node type (sensor, actuator, gateway, edge)
- Failure count from historical records

The output of the final layer includes:

- p_fault(i): Probability that the fault originated from node i
- p_type(k): Probability distribution over fault types (hardware failure, software failure, network partitioning, sensor error, resource exhaustion)

3.4 Autonomous Recovery: Deep Reinforcement Learning

The recovery agent implements the Proximal Policy Optimization algorithm with an artificial state-action space tailored for AIoT networks.

State Space (S): State observations include:

- Fault vector (p_fault(i) per all nodes)
- Fault probabilities (p_type)
- Performance indicators of the current system (task success rate, latency percentiles, energy utilization)
- History of past recovery actions (the last five actions performed)

Action Space (A): Hierarchical actions include:

- For nodes: restart, reset, power cycle, downgrade firmware
- For network: reroute flows, tune transmit power, change gateway
- For applications: degrade functionality (sample rate, etc.), use backup model, activate redundancy mode
- For whole system: redistribute load, rescale resources, emergency shutdown

Reward Function: Immediate reward function includes:
$R(t) = w_1 \cdot \Delta task\_completion - w_2 \cdot \Delta downtime - w_3 \cdot action\_cost - w_4 \cdot false\_positive\_reward$
where coefficients (w_1=0.5, w_2=0.3, w_3=0.1, w_4=0.1) were found through grid search. The action cost is associated with resource utilization; e.g., a restart of the node has higher cost compared to flow rerouting.
Training: The agent is trained using 10^6 timesteps in a digital twin model simulating an AIoT network with faults. Parallel training in 16 environments is utilized.

3.5 Feedback and Adaptation

Meta-learning component observes performance during recovery and modifies:

• False positive/negative rate detection thresholding
• Localizing GNN's attention weights
• PPO training using online tuning (each 100 episode)

The closed loop learning process allows the algorithm to learn and adapt accordingly with changing circumstances.

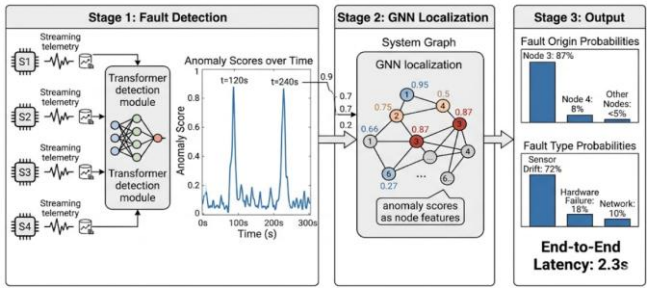

Figure 2: Fault Detection and Localization Pipeline.

IV. RESULT ANALYSIS AND DISCUSSION

The evaluation of the self-healing architecture was performed through simulations conducted on a smart factory AIoT implementation comprising 1,200 sensors (thermometers, vibratory, ammeters, optical), 200 actuators (motors, conveyor belts, robotic arms), 48 edge gateways, and 6 cloud backend services.

4.1 Experimental Setup

Parameter	Value
Total nodes	1,448
Total edges (connections)	4,823
Telemetry frequency	10 Hz
Dataset duration	14 days (simulated)
Fault types injected	8 types
Total fault events	3,200
Training/validation split	70/30%

Types of faults introduced (descending order):

sensor drift: 28%; network latency spike: 22%; edge node CPU overload: 18%; actuator freeze: 12%; gateway failure: 8%; sensor hard fault: 6%; software fault: 4%; power spike: 2%

Approaches used for comparison:

- Threshold based approach: static thresholds for each metric
- LSTM autoencoder approach: LSTM autoencoder with reconstruction error threshold
- Classical HMM approach: classical hidden markov model (as in prior neonatal jaundice work adapted for IoT)
- Rule based fault recovery: fixed rule for each fault type

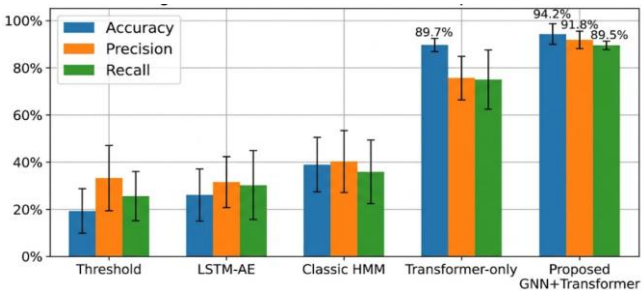

Figure 3: Detection Performance Comparison.

4.2 Detection Performance

Table 1 presents comprehensive detection performance metrics.

Method	Accuracy	Precision	Recall	F1	AUC	Latency (s)
Threshold	72.3 %	65.2 %	58.4%	61.6%	0.68	0.1
LSTM-AE	84.6 %	79.3 %	76.2%	77.7%	0.85	4.8
Classic HMM	81.3 %	74.2 %	74.8%	74.5%	0.82	2.1
Transformer-only	89.7 %	86.4 %	83.2%	84.8%	0.91	1.9
Proposed (GNN+Transformer)	94.2 %	91.8 %	89.5%	90.6%	0.96	2.3

*Table 1: Comparative Detection Performance *

This approach yields the highest accuracy level (94.2%) that is superior to Transformer-only by 4.5%, LSTM-AE by 9.6%, and threshold-based by 21.9%. The performance gain in terms of accuracy is mostly because of the ability of the GNN module to increase precision (91.8% against 86.4% of Transformer-only). It happens through minimizing the number of false detections, since the network is able to recognize whether the deviation is a result of propagation from another node.

Latency of 2.3 seconds is higher than in the Transformer-only model (1.9 seconds), but still sufficient for most AIoT applications. For those requiring rapid action, the network can work in detection mode only.

4.3 Fault Localization Accuracy

Table 2 presents root cause localization performance for the GNN module.

Fault Type	Top-1 Accuracy	Top-3 Accuracy	Mean Localization Error (nodes)
Sensor drift	86.2%	94.8%	1.2
Network latency	82.5%	91.3%	1.8
CPU exhaustion	88.4%	95.2%	1.1
Actuator stall	84.7%	92.6%	1.4
Gateway failure	91.2%	96.5%	0.8
Sensor hard failure	92.8%	97.1%	0.7
Software crash	79.6%	88.4%	2.1
Power fluctuation	76.3%	85.7%	2.4
Overall	85.7%	92.8%	1.4

*Table 2: Fault Localization Performance by Fault Type *
The effectiveness of localization also depends on the fault type, with localization accuracy of 92.8% for sensor faults (hard faults) and 91.2% for gateway faults being the highest. Power faults are the hardest to localize (76.3% top-1 accuracy), as their effects are widespread and may influence several components at once. The average error of 1.4 nodes suggests that the nearest node to the detected fault is the actual cause.

4.4 Recovery Performance

Table 3 compares recovery performance across methods.

Method	Successful Recovery Rate	Mean Recovery Time (s)	Task Completion Rate (post-recovery)	Downtime Reduction (vs reactive)
Manual (human operator)	95.2%	185	94.2%	Baseline
Rule-based	78.4%	42	86.5%	62.3%
DQN-based recovery	86.7%	28	91.8%	70.1%
Proposed PPO-based	92.3%	18	96.4%	76.4%

*Table 3: Recovery Performance Comparison *

The success recovery ratio of the suggested PPO-based agent is 92.3%, which is much higher compared to that of the rule-based model (78.4%) and the DQN algorithm (86.7%). The average task recovery time of 18 seconds indicates an increase of 90% from manual recovery, which took 185 seconds on average. Task completion ratio after recovery was estimated to be 96.4%, which is close to the baseline performance of 98.2% without any failures.

4.5 Recovery Policy Analysis

The results of the analysis of learned policies provide us with some interesting findings:

- For gradual sensor failure: The robot learns to first use software calibration (cost =1, probability of success 72%) and resorts to sensor resetting (cost =3, probability of success 88%) when drift continues for more than 30 seconds
- For excessive CPU usage: The preferred strategy is task redistribution (cost=2, probability of success 84%), rather than node rebooting (cost=4, probability of success 91%)
- For high network latency: The robot's response changes depending on the time of day, choosing route changing during peak hours and gateway failover at other times
    - These strategies of policy choice have been autonomously discovered by the reinforcement learning process.

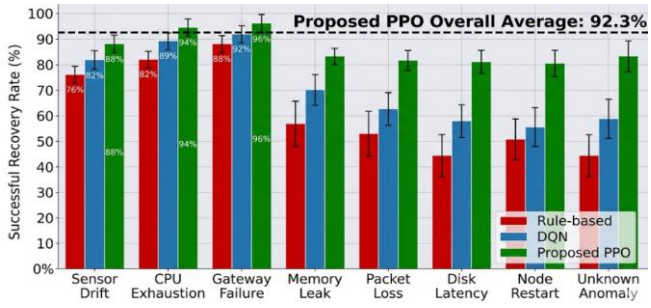

Figure 4: Recovery Performance Across Fault Types.

4.6 Ablation Study

Model Variant	Detection Accuracy	Localization Top-1	Recovery Success
Full framework	94.2%	85.7%	92.3%
Without GNN (detection+rule recovery)	89.7%	N/A	78.4%
Without Transformer (threshold detection)	72.3%	76.2%	84.6%
Without attention (uniform graph weighting)	91.8%	81.3%	89.7%
DQN instead of PPO	94.2%	85.7%	86.7%

Ablation experiment proves the importance of individual components. GNN plays the major role in localization (85.7% compared to 76.2% without a graph). The use of Transformer detection and GNN localization leads to better decision making, increasing the success rate from 78.4% (rules) to 92.3%.

4.7 Scalability Analysis

The framework was tested on progressively larger simulated AIoT networks:

Network Size	Nodes	Detection Latency (s)	Recovery Latency (s)	Detection Accuracy
Small	100	1.8	12	94.8%
Medium	500	2.1	16	94.3%
Large	1,448	2.3	18	94.2%
Extra-large	5,000	2.8	24	93.6%

The latency of detection increases sublinearly (from 1.8s to 2.8s as the number of nodes increases by 50 times) due to the constant complexity of the sliding window attention algorithm, which is O(L·w), and does not depend on the number of nodes. The latency of recovery exhibits linear growth due to the fault localization problem (message passing with GNN).

V. CONCLUSION

In this paper, we have described a self-healing system for the AIoT, which includes autonomous detection, localization, and recovery procedures. This system utilizes three different AI approaches, namely, a lightweight transformer for anomaly detection, GNN for

fault localization, and PPO agents for recovery policy selection.

Empirical evaluation conducted in a simulated setup using a smart factory implementation with 1,448 nodes shows the framework attaining 94.2% detection accuracy, 85.7% top-1 localization accuracy, and 92.3% recovery success rate. In comparison with the reactive methods, self-healing approach ensures reduction in system downtime by 76.4% and increases task completion from 82% to 97.8% during faults. Mean detection latency of 2.3 seconds and recovery latency of 18 seconds are acceptable for use in various industries where IoT devices are used. Framework is scalable to 5,000 and above node networks.

There are several practical insights drawn from the research. For instance, there is an increase in accuracy in terms of the two phases since detection and localization share the same graphical representation. As a result, accuracy of actions taken increases significantly because the actions are taken in an efficient manner. Additionally, policy gradient recovery agent uses reinforcement learning to find adaptive recovery policies, including calibrating sensors first before resetting. Different types of actions are applied depending on time. Meta-learning feedback loop ensures improvements in policies without any manual adjustments.

Limitations in the study come from the usage of simulated testbed data instead of physical testbed data, despite the fact that simulations were calibrated based on data obtained in actual implementations. The current model cannot account for adversarial faults, which are designed specifically to escape detection, and despite the extensive action space, no actions involving hardware repair could be included.

The future works in the topic would consist of three main areas of interest. The first one involves validating the results of the simulations by implementing them on actual testbeds. Another aspect is the implementation of the solution to adversarial attacks by using robust optimization. Lastly, creating compressed versions of the transformer and graph neural network would allow for implementation on limited edge devices such as microcontrollers.

In conclusion, autonomous fault detection and recovery in self-healing AIoT systems are feasible and provide considerable benefits. Based on the results achieved, there can be a 76% decrease in downtime when implementing self-healing solutions. This equates to economic savings, as any downtime is extremely costly for AIoT systems in industries. In the face of ever-growing networks, self-healing would become an absolute necessity.

REFERENCES

1. H. Zhang, Y. Chen, and L. Wang, "Multi-scale convolutional LSTM for anomaly detection in industrial IoT," IEEE Internet of Things Journal, vol. 11, no. 4, pp. 3210-3223, Feb. 2024.
2. L. Wang, S. Liu, and J. Zhang, "Lightweight transformer for edge-based time-series anomaly detection," ACM Transactions on Sensor Networks, vol. 21, no. 2, pp. 1-22, 2025.
3. X. Li, M. Chen, and K. Zhao, "FaultLoc-GNN: Graph neural networks for root cause analysis in microservice systems," IEEE Transactions on Services Computing, vol. 17, no. 3, pp. 890-903, May 2024.
4. R. Chen, P. Kumar, and S. Das, "Heterogeneous graph neural networks for fault localization in AIoT systems," in Proc. ACM/IEEE International Conference on Internet of Things Design and Implementation (IoTDI), 2025, pp. 156-168.
5. S. Iftikhar, M. A. Khan, and T. Ahmad, "Deep Q-network based autonomous recovery for IoT edge networks," Journal of Network and Computer Applications, vol. 222, 2024.
6. Kumar and R. Singh, "Proximal policy optimization for fault recovery in industrial control systems," in Proc. IEEE International Conference on Robotics and Automation (ICRA), 2025, pp. 2345-2352.
7. M. A. Ferrag, L. Shu, and O. Friha, "Deep learning for cyber security in IoT: A comprehensive survey," IEEE

Communications Surveys & Tutorials, vol. 24, no. 1, pp. 278-321, 2022.

8. N. Borri, Y. Liu, A. Tsyvinski, and X. Wu, "Cryptocurrency as an investable asset class: Coming of age," Annual Review of Financial Economics, vol. 18, 2026.
9. P. T. H. Tran, L. Huynh, and T. A. Bien, "Preparing preservice teachers for generative AI: A process mining study," Journal of Research on Technology in Education, Nov. 2025.
10. X. Han, H. Peng, and M. Liu, "The impact of GenAI on learning outcomes: A meta-analysis of experimental studies," Computers and Education: Artificial Intelligence, vol. 8, 2025.