MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

Multi-Hop IoT Network Fault Detection Using Spatio-Temporal Graph Neural Network

Lakha, Bishal; Guo, Jianlin; Parsons, Kieran; Sumi, Takenori; Nagai, Yukimasa; Serra, Edoardo

TR2026-059 May 28, 2026

Abstract

Large-scale multi-hop wireless IoT networks are increasingly being deployed to support critical applications with stringent QoS requirements. Robust fault detection in such networks is vital for ensuring normal network operations and enabling proactive maintenance especially in adversarial environments. However, the existing researches primarily focus on data- driven attack detection, leaving natural network fault detection largely unexplored. Due to the absence of dedicated routers, multi-hop IoT networks require network nodes to transmit their own data and also relay data for others, which introduces inherent and cascaded anomalies, making it difficult to locate anomaly sources. Furthermore, the underlying CSMA/CA based communication protocols adopted in IoT networks incur un- predictable delays in transmissions, thereby complicating timely anomaly detection and accurate diagnosis. This paper proposes innovative network fault detection technologies tailored for large- scale multi-hop IoT networks. We introduce a spatio-temporal graph neural network (STGNN) model capable of accurately detecting both node-level and edge-level faults. Our model takes both temporal data and metadata for context-aware forecasting and reconstruction. To address the lack of labeled data, we adopt an unsupervised learning paradigm. We evaluated our model on various large-scale network topologies and transaction log datasets, and results demonstrate that our model consistently outperforms baseline models across most cases.

IEEE International Conference on Communications Workshops (ICC) 2026

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

Multi-Hop IoT Network Fault Detection Using Spatio-Temporal Graph Neural Network

Bishal Lakha [*§], Jianlin Guo[*], Kieran Parsons[*], Takenori Sumi[†], Yukimasa Nagai[‡], Edoardo Serra [§]
[*]Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA 02139, USA
[†]Information Technology R&D Center, Mitsubishi Electric Corporation, Kamakura, Kanagawa 2478501, Japan
[‡]IoT Life Solution Business Strategy Center, Mitsubishi Electric Corporation, Yokohama, Kanagawa, Japan
[§]Department of Computer Science, Boise State University, Boise, ID 83725, USA

***Abstract*—Large-scale multi-hop wireless IoT networks are increasingly being deployed to support critical applications with stringent QoS requirements. Robust fault detection in such networks is vital for ensuring normal network operations and enabling proactive maintenance especially in adversarial environments. However, the existing researches primarily focus on data-driven attack detection, leaving natural network fault detection largely unexplored. Due to the absence of dedicated routers, multi-hop IoT networks require network nodes to transmit their own data and also relay data for others, which introduces inherent and cascaded anomalies, making it difficult to locate anomaly sources. Furthermore, the underlying CSMA/CA based communication protocols adopted in IoT networks incur unpredictable delays in transmissions, thereby complicating timely anomaly detection and accurate diagnosis. This paper proposes innovative network fault detection technologies tailored for large-scale multi-hop IoT networks. We introduce a spatio-temporal graph neural network (STGNN) model capable of accurately detecting both node-level and edge-level faults. Our model takes both temporal data and metadata for context-aware forecasting and reconstruction. To address the lack of labeled data, we adopt an unsupervised learning paradigm. We evaluated our model on various large-scale network topologies and transaction log datasets, and results demonstrate that our model consistently outperforms baseline models across most cases.**

***Index Terms*—Fault detection, multi-hop IoT network, spatio-temporal graph neural network, unsupervised learning.**

I. Introduction

The Internet of Things (IoT) has become a foundational technology driving mission-critical applications such as smart utility. At scale, these deployments commonly form multi–hop networks in which resource constrained nodes forward one another's data traffic over contention based wireless links (e.g., IEEE 802.15.4 link and IEEE 802.11 link), coordinated by routing protocols such as widely used IETF IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL).

While this architecture enhances scalability and coverage, it also introduces significant challenges for reliability and resilience. Accordingly, fault detection in such networks is crucial to ensure normal network operations and enable proactive maintenance especially in adversarial environments. However, fault detection in multi-hop IoT networks faces significant challenges. IoT networks typically do not have dedicated router. As a result, network nodes not only transmit their own data but also relay data for other nodes, which can cause fault propagation phenomena. Fault in single node or edge can result in fault for all upstream nodes in routing topology. In addition, the random backoff in CSMA/CA based communication protocols such as IEEE 802.15.4 and IEEE 802.11 incurs additional challenges including random transmission delay and collision.

Treating faults as anomalies offers a principled way to model the open-ended, context-dependent deviations that arise in multi-hop IoT networks. In this view, a "fault" is any statistically rare departure from a nominal spatio-temporal behavior of a node or an edge, rather than a pre-enumerated error class. This framing enables unsupervised/self-supervised learning on abundant unlabeled operational data, naturally surfaces previously unseen (zero-day) failure modes, and reduces costly labeling and fault-specific feature engineering [1].

Multi–hop IoT networks naturally induce time–varying communication graphs. In such networks, treating each node as an independent time series or applying flat, tabular outlier detectors ignore this relational structure, which is known to inflate false alarms and miss *collective* or *contextual* anomalies that manifest only through coordinated deviations over nodes, edges and subgraphs. Spatio-temporal graph neural models can encode these dependencies by combining message passing over the network topology with temporal dynamics, yielding superior forecasting and detection in time-dependent settings while supporting node/edge/subgraph-level topology.

The current literature on graph-based IoT network anomaly detection tends to bifurcate into (i) *intrusion/attack detection* [2] and (ii) *application/sensor–value modeling* that flags outliers directly in physical measurements, instead of network logs and metadata, and mostly in single-hop IoT network [3], [4]. But, beyond adversarial behavior, naturally occurring faults in multi-hop IoT networks are pervasive and comparably disruptive even in well-engineered deployments [5]. Using network logs and metadata for anomaly detection is application-agnostic, privacy-preserving and robust to encryption, storage and compute-efficient, and often provides earlier precursors of cascading failures than payload/sensor deviations [6].

In this work, we make several key contributions towards advancing fault detection in large-scale multi-hop IoT networks. (1) We formulate fault detection as a probabilistic anomaly prediction task using a Gaussian negative log-likelihood (NLL) objective, which explicitly accounts for the inherent randomness and noise in contention based wireless communication. (2) We develop an unsupervised and a unified spatio-temporal graph neural network (STGNN) that jointly detects node-level faults (e.g., battery depletion, data corruption) and edge-level faults (e.g., link degradation, edge breakage) by leveraging

This work was done while Bishal Lakha was working at Mitsubishi Electric Research Laboratories (MERL) as an intern.

both temporal network transaction logs and network topology metadata. (3) We introduce four large-scale multi-hop IoT network fault detection datasets capturing realistic network dynamics—stochastic transmission backoff, link variability, and interference—to support research in this unexplored domain.

II. Related Work

Data-driven anomaly detection in IoT networks has attracted significant research attention in recent years. For instance, paper [7] proposes an efficient unsupervised framework for anomaly detection over edge-assisted IoT, where data from single-hop sensor networks are aggregated by an edge device for fault identification. Similarly, paper [8] applies the Isolation Forest algorithm for autonomous fault detection and recovery in self-healing IoT sensor networks. Recent work [9] introduces a brain-like cognition-driven model factory that combines large language models (LLMs) with lightweight models for industrial IoT fault diagnosis. In a related direction, work [10] proposes a scalable log-based failure diagnosis method built on Retrieval-Augmented Generation (RAG). By leveraging LLM-based summarization, sample augmentation, and Chain-of-Thought (CoT) prompting, it enhances interpretability and efficiency in log-based anomaly analysis.

Beyond traditional and LLM-based systems, several studies have explored the use of graph-based models for capturing relational dependencies in IoT data. For example, work [4] introduces a dynamic-edge graph attention mechanism for early-stage fault detection in industrial IoT systems. The method learns the temporal evolution of inter-sensor relationships through adaptive edge inference, integrating contextual operating conditions into node dynamics to improve detection robustness. Building on similar motivations, paper [11] presents an energy-efficient GNN-based anomaly detection approach for multivariate IoT time series, where inter-sensor correlations are learned via graph structure learning.

Comprehensive surveys such as works [12] and [13] provide valuable overviews of recent advancements in graph-based time-series anomaly detection. Specifically, paper [12] highlights how graph representations can model temporal dependencies to facilitate anomaly detection, reviewing state-of-the-art GNN architectures and identifying open challenges in this emerging field. Meanwhile, paper [13] systematically categorizes deep Graph Anomaly Detection (GAD) methods along three key dimensions: GNN backbone design, proxy task formulation, and graph anomaly measures.

Despite these advances, most existing works overlook the influence of the *physical network topology* on fault detection. In real-world multi-hop IoT networks, factors such as node location, routing paths, node condition, link quality, and traffic patterns provide rich contextual cues that can enhance anomaly detection accuracy and interpretability. To address this gap, our work introduces a GNN-based network fault detection framework for large-scale multi-hop IoT networks that explicitly incorporates physical topology. By embedding node-level attributes (e.g., traffic volume, number of upstream nodes, routes, and neighbors), edge-level metrics (e.g., link quality, endpoint properties, and interference conditions), and network-level features (e.g., routing and communication protocols, network scale), our approach captures both structural and operational characteristics crucial for robust fault diagnosis.

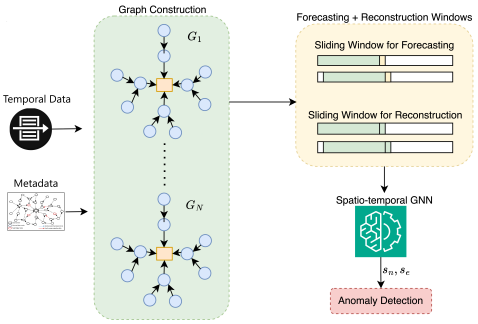


Fig. 1: Pipeline for Network Fault Detection

III. Methodology

Our IoT network fault detection pipeline, shown in Fig. 1, ingests network transaction logs as temporal data and physical network topology information as metadata to construct dynamic graphs. The proposed model performs joint forecasting and reconstruction over spatio-temporal graph sequences using a Spatio-Temporal Graph Neural Network (STGNN). The model is trained exclusively on normal operational data to learn typical network behavior. Deviations between the predicted and observed attributes yield node-level and edge-level anomaly scores, which are then used for fault detection.

A. Network Topology Based Graph Construction

To embed physical network topology information into multi-hop network fault detection, we propose graph-based fault detection technologies. Accordingly, we consider an ordered multiset of network transaction logs formatted as

$$\mathscr{L} = \left\{\ell_i = \left(t_i, n_i, c_i, s_i, b_i, h_i, lq_i\right)\right\}_{i=1}^{M},$$

where each entry ℓ_i in log dataset $\mathscr{L}$ corresponds to a single transaction record in the system log. The logs are temporally ordered, thereby preserving the causal progression of events in the underlying network. This ordering is essential, as it enables subsequent temporal analysis and the recovery of traffic flows across space and time. Specifically, the components of ℓ_i are defined as follows:

t_i	$\in \mathbb{R}_{\geq 0}$	timestamp (millisecond)
n_i	$\in V = \{0, \ldots, n\}$	node that generated the log (**0** = data center)
c_i	$\in \{0, 1, 2\}$	transaction code: $0 =$ TX, $1 =$ RX, $2 =$ Relay
s_i	$\in V$	original source of the packet
b_i	$\in \mathbb{N}$	payload size (bytes)
h_i	$\in V$	*next-hop* if $c_i \in \{0,2\}$; *previous-hop* if $c_i = 1$
lq_i	$\in \mathbb{R}_{\geq 0}$	link quality, when $c_i = 1$

This schema captures both the structural and operational aspects of network traffic flows. Transmission and relay operations encode directed edges, while reception operations enrich the edge attributes with link quality information, enabling a fine-grained characterization of network health.

1) Time Discretization: Continuous transaction logs are transformed into a temporally consistent representation to

enable graph-based modeling. We partition time into uniform intervals of length Δ, defining time slice boundaries as

$$T_k = k\Delta, \quad k = 0,1,\ldots,N, \qquad N = \left\lfloor \frac{\max\{t_i\}}{\Delta} \right\rfloor.$$

The k_{th} data subset then consists of all events

$$\mathscr{L}_k = \{\ell_i \in \mathscr{L} : T_k \le t_i < T_{k+1}\}, \quad T_{k+1} = T_k + \Delta.$$

This discretization transforms a raw event log into a sequence of temporally aligned graph snapshots, each reflecting the network state within a fixed observation horizon.

2) Graph Snapshot G_k: For each slice k, we construct a directed, attributed graph snapshot G_k which is defined over the set of active vertices:

$$V_k = \{v \mid v \text{ generates log in } \mathscr{L}_k\}.$$

Directed edges are then formed by mapping transmission, reception and relay events: for each log $\ell_i \in \mathscr{L}_k$ with $c_i \in \{0,2\}$, we add the directed pair $(u,v) = (n_i, h_i)$ and for each log $\ell_i \in \mathscr{L}_k$ with $c_i = 1$, we add the directed pair $(u,v) = (h_i, n_i)$, resulting in edge set:

$$E_k = \{(u,v) \mid \text{either } u \text{ or } v \in V_k\} \subseteq \bar{V}_k \times \bar{V}_k,$$

where $\bar{V}_k = V_k + \{h_i\}$. This construction jointly captures the active communication endpoints and the flow of packets during the time interval $[T_k, T_{k+1})$.

a) Node-feature matrix $\mathbf{X}_k \in \mathbb{R}^{|V_k|\times 6}$: For each node $v \in V_k$, we aggregate transmission, reception, and relay statistics:

$$\text{tx\_pkt}_v^k = \sum_{\mathscr{L}_k : n_i = v, c_i = 0} 1, \qquad \text{tx\_byte}_v^k = \sum_{\mathscr{L}_k : n_i = v, c_i = 0} b_i,$$

$$\text{rx\_pkt}_v^k = \sum_{\mathscr{L}_k : n_i = v, c_i = 1} 1, \qquad \text{rx\_byte}_v^k = \sum_{\mathscr{L}_k : n_i = v, c_i = 1} b_i,$$

$$\text{rl\_pkt}_v^k = \sum_{\mathscr{L}_k : n_i = v, c_i = 2} 1, \qquad \text{rl\_byte}_v^k = \sum_{\mathscr{L}_k : n_i = v, c_i = 2} b_i.$$

We then convert the counts to rates and therefore, define node feature vector $\mathbf{x}_v^k$ as

$$\mathbf{x}_v^k = \tfrac{1}{\Delta}(\text{tx\_pkt}_v^k, \text{tx\_byte}_v^k, \text{rx\_pkt}_v^k, \text{rx\_byte}_v^k, \text{rl\_pkt}_v^k, \text{rl\_byte}_v^k)^\top.$$

To mitigate degree-related biases, we normalize node features using the ancestor set size, where the ancestor set $\mathscr{A}_v^k$ of node v at slice k is the upstream subtree rooted at v. Specifically, it contains all the nodes that can reach v through a directed path in routing topology. Formally,

$$\mathscr{A}_v^k = \{u \in V : \text{there exists a directed path } u \to \cdots \to v \text{ for } v \in V_k\}.$$

For the feature vector x_v^k of node v at slice k, the normalized feature vector is given as: $\tilde{x}_v^k = \frac{x_v^k}{|\mathscr{A}_v^k|+1}$. The final node feature matrix is given as $\mathbf{X}_k = \{\tilde{x}_v^k\}$.

b) Edge-feature matrix $\mathbf{E}_k \in \mathbb{R}^{|E_k|\times 3}$: For every directed edge $(u,v) \in E_k$, we aggregate:

$$(\text{fwd\_pkt}_{(u,v)}^k, \text{fwd\_byte}_{(u,v)}^k) = \sum_{\substack{\ell_i \in \mathscr{L}_k: \\ n_i = u, h_i = v, c_i \in \{0,2\}}} (1, b_i)$$

$$w_{(u,v)}^k = \begin{cases} \dfrac{1}{\ell_{\max}(|\mathscr{R}_{(u,v)}^k| + |\mathscr{N}_v|)} \displaystyle\sum_{\ell_i \in \mathscr{R}_{(u,v)}^k} lq_i, & |\mathscr{R}_{(u,v)}^k| > 0, \end{cases}$$

where $\mathscr{R}_{(u,v)}^k := \{\ell_i \in \mathscr{L}_k \mid u = h_i,\ v = n_i,\ c_i = 1,\ s_i \in \mathscr{A}_v^k\}$ collects all reception events at slice k that report the forward link $(u \to v)$, $\mathscr{N}_v$ is the set of node v's physical communication neighbors that can interfere with the link $(u \to v)$, and $\ell_{\max}$ is the maximum link quality and its value depends on link quality metric and its implementation, e.g., 255 for RSSI. The first two features encode traffic intensity, while the third feature provides a compact proxy for physical link reliability. Together, they allow G_k to simultaneously reflect logical connectivity and physical channel conditions. Finally, each edge vector $\mathbf{e}_{(u,v)}^k \in \mathbf{E}_k$ is given as

$$\mathbf{e}_{(u,v)}^k = \left(\tfrac{1}{\Delta}\,\text{fwd\_pkt}_{(u,v)}^k,\ \tfrac{1}{\Delta}\,\text{fwd\_byte}_{(u,v)}^k,\ w_{(u,v)}^k\right).$$

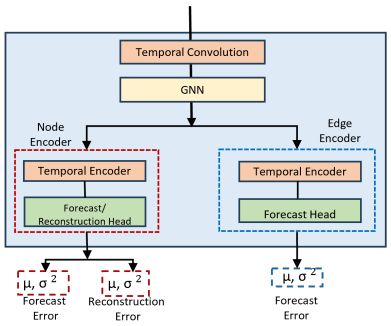


Fig. 2: Block diagram of Proposed IoT Network Fault Detector

c) Graph snapshot and temporal graph: A graph snapshot is an attributed graph capturing network activity during one time slice, given as:

$$G_k = (V_k, E_k, \mathbf{X}_k, \mathbf{E}_k), \qquad \forall k \in [0,N].$$

The chronologically ordered sequence of such snapshots is the discrete temporal graph we use for our fault detection task.

B. Fault Detection Problem Formulation

We formulate the fault detection task as node and edge attribute prediction and reconstruction over a *sliding window*. At each step, we use the most recent h graph snapshots to predict the next snapshot and reconstruct the last snapshot for nodes and predict the next snapshot for edges as shown in Fig. 1. We detail the approach as follows.

1) Forecasting and reconstruction window: For history length $h \ge 1$, we define the length-h forecast window as

$$\mathbf{F}_k = (G_{k-h+1}, \ldots, G_k), \qquad k \ge h.$$

We use a stride of 1, so windows progress by dropping the oldest snapshot and appending the newest snapshot as

$$\mathbf{F}_{k+1} = (G_{k-h+2}, \ldots, G_k, G_{k+1}).$$

At each step k,

- **Forecasting** targets attributes of the *next* snapshot $G_{k+1} = (V_{k+1}, E_{k+1}, \mathbf{X}_{k+1}, \mathbf{E}_{k+1})$, i.e., $\mathbf{X}_{k+1}$ and $\mathbf{E}_{k+1}$.
- **Reconstruction** targets node attributes of the *current* snapshot G_k, i.e., $\mathbf{X}_k$.

2) Spatio-Temporal GNN model: Given the forecast window $\mathbf{F}_k$, we learn a mapping that (i) forecasts the next snapshot's node/edge attributes and (ii) reconstructs the most recent node attributes:

$$(\widehat{\mathbf{X}}_{k+1}, \widehat{\mathbf{E}}_{k+1}, \widetilde{\mathbf{X}}_k) = \mathscr{F}(\mathbf{F}_k),$$

where $\mathscr{F}$ is the proposed Spatio-Temporal Graph Neural Network (STGNN) model shown in Fig. 2, which consists of a shared spatio-temporal block followed by node-specific and edge-specific heads. Our model has four components:

a) Temporal Convolution: For every entity a (node v or edge (u,v)), we apply a causal, dilated 1-D temporal convolution to its feature sequence inside the window to capture short-term dynamics.

b) Graph Neural Network: At each slice k, the temporally convolved features $\mathbf{h}_{node}^{(0)}$ and $\mathbf{h}_{edge}^{(0)}$ are propagated over the directed graph G_k with n message-passing layers message$^{(k)}$, aggregate$^{(k)}$, and update$^{(k)}$:

$\mathbf{m}_{u\to v}^{(k)} = \text{message}^{(k)}\big(\mathbf{h}_u^{(k-1)}, \mathbf{h}_v^{(k-1)}, \mathbf{h}_{(u,v)}^{(k-1)}\big)$,

$\bar{\mathbf{m}}_v^{(k)} = \text{aggregate}^{(k)}\big\{\mathbf{m}_{u\to v}^{(k)} : u \in \mathscr{N}^-(v)\big\}$,

$\mathbf{h}_v^{(k)} = \text{update}^{(k)}\big(\mathbf{h}_v^{(k-1)}, \bar{\mathbf{m}}_v^{(k)}\big)$, where set $\mathscr{N}^-(v)$ contains nodes that form directed edges towards node v. By choosing concrete realizations for message, aggregate, and update, we get different GNN families (GraphSAGE, GIN, GAT), where GNN in our STGNN is GAT based. This stage mixes time-local features across current network topology so that representations reflect both local history and neighborhood context.

c) Node Encoder: Each node v aggregates its sequence of backbone embeddings across the window via a node temporal encoder TempEnc$^{(v)}$, then two MLP heads produce a one-step-ahead forecast and a reconstruction of the last slice:

$$\mathbf{g}_k^{(v)} = \text{TempEnc}^{(v)}\big(\mathbf{h}_v^{(k-h+1)}, \ldots, \mathbf{h}_v^{(k)}\big),$$

$$\widehat{\mathbf{x}}_v^{k+1} = \text{MLP}_{\text{f}}^{(v)}(\mathbf{g}_k^{(v)}),$$

$$\widetilde{\mathbf{x}}_v^{k} = \text{MLP}_{\text{r}}^{(v)}(\mathbf{g}_k^{(v)}),$$

where TempEnc$^{(v)}$ can be model such as LSTM or Informer.

d) Edge Encoder: Analogously for each directed edge (u,v), we use an edge temporal encoder TempEnc$^{(e)}$ to capture longer-range interaction patterns like persistent traffic intensity and link-quality trends:

$$\mathbf{g}_{(u,v),k}^{(e)} = \text{TempEnc}^{(e)}\big(\mathbf{h}_{(u,v)}^{(k-h+1)}, \ldots, \mathbf{h}_{(u,v)}^{(k)}\big),$$

where TempEnc$^{(e)}$ can also be model such as LSTM or Informer. We then use a single forecasting head to predict the next-step edge attributes:

$$\widehat{\mathbf{e}}_{(u,v)}^{k+1} = \text{MLP}_{\text{f}}^{(e)}(\mathbf{g}_{(u,v),k}^{(e)}).$$

In the edge encoder, we only stick with the forecasting head and omit the reconstruction head as edge signals are largely determined by adjacent nodes' data flows in the network topology, so a separate reconstruction objective is mostly redundant with the node reconstruction loss.

3) Training loss.: We adopt a heteroscedastic weighted Gaussian negative log-likelihood (NLL) as the primary objective. For a D-dimensional target vector y with predicted mean μ and variance σ^2, the loss is defined as:

$$L_{\text{NLL}}(y,\mu,\sigma) = \frac{1}{2}\sum_{d=1}^{D} w_d \left[\frac{(y_d-\mu_d)^2}{\sigma_d^2} + 2\ln\sigma_d + \text{const}\right],$$

where $w_d \geq 0$ is a learned attribute-specific weight in training.

Multi-hop IoT networks exhibit inherent noise and randomness due to traffic burstiness, channel access contention, interference, and asynchronous reporting. A fixed-noise loss (e.g., MSE) implicitly assumes homoscedastic errors and tends to over-penalize high-variance attributes (busy nodes/edges), while under-penalizing low-variance ones (quiet nodes/edges). The heteroscedastic loss L_{NLL} addresses this by *learning* an uncertainty σ_d per attribute: (i) the scaled error term $(y_d-\mu_d)^2/\sigma_d^2$, denoted as ε, down-weights inherently noisy signals, and (ii) the $\ln\sigma_d$ term prevents the model from inflating σ_d to escape errors, yielding well-calibrated uncertainties. Consequently, the model focuses on *systematic* deviations rather than stochastic fluctuations, improving generalization and producing anomaly scores that are naturally normalized across attributes and operating conditions.

The overall training objective combines forecasting and reconstruction losses based on L_{NLL} given as:

$$L_{node} = w_f L_f^n + (1-w_f)L_r^n,$$

$$L_{\text{total}} = w_f L_f^n + (1-w_f)L_r^n + w_e L_f^e,$$

where L_f^n and L_r^n denote node-level forecasting and reconstruction losses, respectively, while L_f^e is the edge-level forecasting loss. The hyperparameters $w_f \in [0,1]$ balance the relative contributions of these terms. This ensures that both accuracy and structural fidelity are simultaneously optimized. Similarly, w_e balances the contribution of edge loss to overall loss.

C. Fault detection.

Once the model is trained, we quantify faults by how *deviated* the observed attributes are from their model-implied values. The node and edge fault scores at slice k are given as:

$$s_v^k = w_f\varepsilon_f(v) + (1-w_f)\varepsilon_r(v),\ v \in V_k,$$

$$s_e^k = \varepsilon_f(e),\ e \in E_k.$$

A node fault is declared whenever $s_v^k > \tau_v$ for a node fault threshold τ_v. Similarly, an edge fault is declared whenever $s_e^k > \tau_e$ for an edge fault threshold τ_e. One approach for determining thresholds τ_v and τ_e can use a *percentile-based approach* on scores computed from benign (fault-free) data. Specifically, one can estimate these thresholds as empirical percentiles (e.g., 95th or 99th) of the benign score distributions [14], [15]. Once learned, the thresholds are fixed and applied during testing to flag anomalous nodes and edges.

IV. Dataset and Anomalies

No public dataset found fits the needs of multi-hop network fault detection. Therefore, we apply NS3 trace files as network log datasets, in which node faults and edge faults are randomly generated. Node faults includes node breakdowns emulating hardware failure and packet error emulating attacking or software failure. Edge faults include low link quality emulating link degradation. Each data generation runs for 10 hours. IEEE 802.15.4 is used as communication protocol and IETF RPL is adopted as network protocol to build routing topology.

a) Circle node deployment: 50 and 250 nodes are deployed in circles with radius of 600m and 1600m, respectively, via Sun-Flower algorithm with data centers at center. Networks normally operate for the first 7 hours. 26 random faults for 50 nodes (8 node breakdowns, 3 packet errors and 15 edge faults) and 159 random faults for 250 nodes (94 node breakdowns, 16 packet errors and 49 edge faults) start from the 8-th hour.

b) Square node deployment: 50 and 250 nodes are randomly deployed in 600m$\times$600m and 1600m$\times$1600m squares, respectively, with data centers at corner. This deployment has higher density than circular deployment and therefore, depicts more complicate network environments. Networks normally operate for the first 7 hours. 23 random faults for 50 nodes (8 node breakdowns, 3 packet errors and 12 edge faults) and 214 random faults for 250 nodes (96 node breakdowns, 15 packet errors and 103 edge faults) start from the 8-th hour.

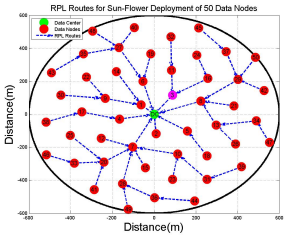


Fig. 3: Routing Topology.

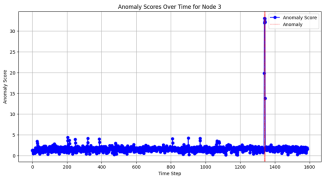


Fig. 4: Node Fault Score.

Fig. 3 shows RPL routing topology for 50-node circle deployment, and Fig. 4 illustrates node fault score over time for magenta node 3 shown in Fig. 3, in which node 3 broke down for 30s, resulting in fault score hikes.

V. Performance Evaluation

We evaluated our GAT based network fault detection model STGNN against widely used anomaly detection baselines including Elliptic Envelope (Elliptic), Local Outlier Factor (LOF), Isolation Forest (IF), and One-Class SVM (OCSVM). All models are trained for both node and edge fault detection. Performance is measured using Area Under ROC Curve (AUROC) and Area Under Precision-Recall Curve (AUPRC), which jointly capture the tradeoff between detection accuracy and robustness under class imbalance without using a threshold. All models are trained under identical conditions with early stopping and hyperparameters tuned via validation.

A. AUROC and AUPRC Scores

Table I shows AUROC and AUPRC scores for all models across four datasets. For node fault detection, our GAT+LSTM achieves the best performance, outperforming next best baseline model by up to 14.8% on AUROC score and 134.6% on AUPRC score, a more important measurement. For edge fault detection of 50-node deployments, our GAT+Informer or GAT+LSTM has similar performance on both AUROC and AUPRC scores as OCSVM, the best baseline model. However, as network size becomes larger for 250 nodes, our GAT+LSTM outperforms the next best baseline model by up to 1.7% on AUROC score and 81.9% on AUPRC score.

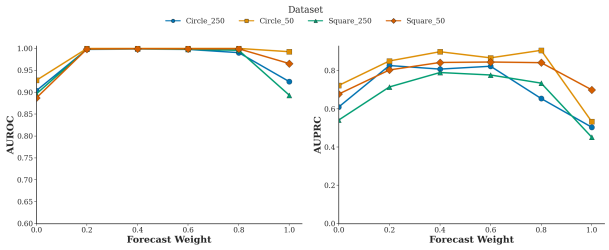
B. Hyperparameter Tuning

We evaluated the impact of key hyperparameters on node fault detection, focusing on the forecasting loss weight (w_f), forecast window length (h), and snapshot size (Δ). Both AUROC and AUPRC peak when $w_f \in [0.2, 0.6]$ (Fig. 5(a)), confirming that a balanced combination of forecasting and reconstruction losses yields optimal performance. Increasing h improves temporal modeling up to 12–24 steps (Fig. 5(b)), while excessively large snapshots ($\Delta > 30s$) reduce sensitivity to short-term anomalies (Fig. 5(c)).

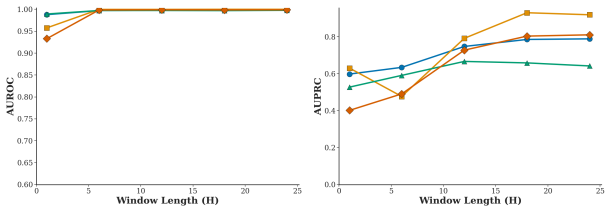
Additional parameters were tuned for stable training and generalization. A learning rate of 1×10^{-3} for the smaller datasets and 5×10^{-4} for the larger datasets ensured smooth convergence, while a small weight decay (5×10^{-4}) mitigated overfitting. A dropout rate of 0.1 and multi-head attention (2–8 heads) provided regularization and improved representation learning. These settings consistently achieve high AUROC and AUPRC across both circle and square datasets.

TABLE I: Performance Comparison with Baselines

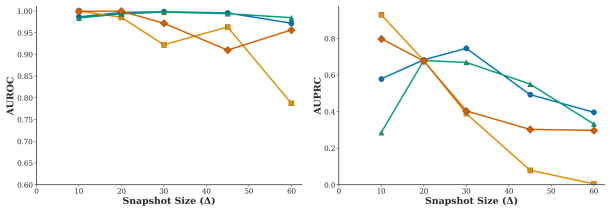
Dataset	Model	Node Fault		Edge Fault	
		AUROC	AUPRC	AUROC	AUPRC
Circle_{50}	Elliptic	0.8696	0.0017	0.8613	0.5149
	LOF	0.8869	0.4741	0.8936	0.6086
	IF	0.9465	0.2279	0.9732	0.4028
	OCSVM	0.8693	0.6373	0.9992	**0.8465**
	GAT+Informer	0.9971	0.7619	**0.9994**	0.7424
	GAT+LSTM	**0.9999**	**0.8863**	0.9881	0.5178
Square_{50}	Elliptic	0.8572	0.0042	0.8650	0.5090
	LOF	0.9938	0.4543	0.7560	0.5023
	IF	0.9313	0.6674	0.9539	0.5253
	OCSVM	0.9285	0.6994	**0.9966**	0.5716
	GAT+Informer	0.9975	0.7259	0.9879	0.7489
	GAT+LSTM	**0.9990**	**0.8411**	0.9909	**0.7516**
Circle_{250}	Elliptic	0.6721	0.0112	0.8633	0.2234
	LOF	0.8698	0.1387	0.6020	0.0888
	IF	0.8556	0.3474	0.9730	0.3942
	OCSVM	0.6723	0.3210	0.4892	0.2987
	GAT+Informer	0.9867	0.7199	0.9867	0.6927
	GAT+LSTM	**0.9987**	**0.8148**	**0.9891**	**0.7171**
Square_{250}	Elliptic	0.7498	0.0092	0.8924	0.2495
	LOF	0.8779	0.2710	0.5873	0.1038
	IF	0.8792	0.2683	0.9516	0.3624
	OCSVM	0.7981	0.3632	0.9746	0.6797
	GAT+Informer	0.9978	0.6728	0.9822	0.7388
	GAT+LSTM	**0.9989**	**0.7894**	**0.9900**	**0.7532**



(a) Forecasting weight (w_f) Vs Node fault performance.



(b) Window length (h) Vs Node fault performance.



(c) Snapshot size (Δ) Vs Node fault performance.

Fig. 5: Hyperparameters Impact on Node Fault Detection

C. Ablation Study

Table II compares the performance of different GNN and temporal encoder (TempEnc) combinations on both node-level and edge-level fault detection across all datasets. Overall, GAT consistently outperforms GIN and GraphSAGE when combined with either the Informer or LSTM temporal encoder. In particular, GAT+LSTM achieves the highest AUROC and

TABLE II: Comparison of GNN+Temporal Models

Dataset	GNN	TempEnc	Node Fault		Edge Fault	
			AUROC	AUPRC	AUROC	AUPRC
Circle_{50}	GIN	Informer	0.9819	0.3436	0.9937	0.5510
	GIN	LSTM	0.9871	0.4508	0.9989	0.3517
	GraphSAGE	Informer	0.9952	0.4947	0.9960	0.5503
	GraphSAGE	LSTM	0.9636	0.4230	0.9967	0.3091
	GAT	Informer	0.9971	0.7619	**0.9994**	**0.7424**
	GAT	LSTM	**0.9999**	**0.8863**	0.9881	0.5178
Square_{50}	GIN	Informer	0.8510	0.2495	**0.9946**	0.4813
	GIN	LSTM	0.9307	0.3698	0.9856	0.1016
	GraphSAGE	Informer	0.9551	0.3390	0.9798	0.7489
	GraphSAGE	LSTM	0.9357	0.3813	0.9857	0.1024
	GAT	Informer	0.9975	0.7259	0.9879	0.7489
	GAT	LSTM	**0.9990**	**0.8411**	0.9909	**0.7516**
Circle_{250}	GIN	Informer	0.9897	0.5849	0.9886	0.7157
	GIN	LSTM	0.9750	0.6452	0.9913	0.7250
	GraphSAGE	Informer	0.9919	0.5804	0.9891	0.7178
	GraphSAGE	LSTM	0.9939	0.7079	**0.9921**	**0.7264**
	GAT	Informer	0.9867	0.7199	0.9867	0.6927
	GAT	LSTM	**0.9987**	**0.8148**	0.9891	0.7171
Square_{250}	GIN	Informer	0.9924	0.5230	0.9862	0.7418
	GIN	LSTM	0.9809	0.4797	0.9838	0.7376
	GraphSAGE	Informer	0.9914	0.4171	0.9835	0.7410
	GraphSAGE	LSTM	0.9960	0.6117	**0.9902**	0.7491
	GAT	Informer	0.9978	0.6728	0.9822	0.7388
	GAT	LSTM	**0.9989**	**0.7894**	0.9900	**0.7532**

AUPRC scores across most datasets, demonstrating its superior capability to capture both spatial and temporal dependencies for detecting network faults. The results also indicate that models with the Informer encoder generally perform competitively, but LSTM tends to yield more stable and higher precision-recall performance, especially in larger graphs

Table III presents the ablation results comparing node fault detection performance with and without key components. For the node encoder, removing either the forecasting or reconstruction head leads to a clear performance drop in both AUROC and AUPRC, confirming that the fault score depends on the joint contribution of both components. The forecasting head enhances temporal sensitivity, while the reconstruction head improves structural consistency, and omitting either weakens detection robustness. In contrast, introducing the forecasting *edge encoder* under multi-task training with the node encoder mostly does not affect performance: AUROC changes are negligible and AUPRC shifts are small, i.e., usually slight improvements and occasionally minor regressions. This suggests that the proposed multi-task learning framework is stable and can integrate edge-level objectives without compromising node performance.

VI. Discussion and Conclusion

Fault detection is crucial for industrial IoT networks to ensure normal network operations and enable proactive maintenance. However, fault detection in multi-hop wireless IoT networks faces significant challenges. This paper presents innovative fault detection technologies for large-scale multi-hop wireless IoT networks. We introduce a spatio-temporal graph neural network (STGNN) model capable of accurately detecting both node-level and edge-level faults. We comprehensively evaluated our STGNN model under various large-scale network deployments. Compared with baseline models, for node fault detection, our model can improve AUROC score by up to 14.8% and AUPRC score by up to 134.6%. For edge fault detection, our model can improve AUROC score by up to 1.7% and AUPRC score by up to 81.9%. The proposed technologies establish a foundation for robust fault detection in noisy, large-scale, and adversarial IoT environments.

TABLE III: Ablation Study on Node Forecasting Head, Node Reconstruction Head, Edge Forecasting Head ($\delta = w/ - w/o$)

Dataset	Component	AUROC			AUPRC		
		w/o	w/	δ	w/o	w/	δ
Circle_{50}	Forecast (Node)	0.9271	0.9999	0.0728	0.7224	0.8863	0.1639
Square_{50}	Forecast (Node)	0.8869	0.9990	0.1121	0.6773	0.8411	0.1638
Circle_{250}	Forecast (Node)	0.9034	0.9987	0.0953	0.6098	0.8148	0.2050
Square_{250}	Forecast (Node)	0.8960	0.9989	0.1029	0.5415	0.7894	0.2479
Circle_{50}	Recon. (Node)	0.9925	0.9999	0.0074	0.5328	0.8863	0.3535
Square_{50}	Recon. (Node)	0.9654	0.9990	0.0336	0.6993	0.8411	0.1418
Circle_{250}	Recon. (Node)	0.9241	0.9987	0.0746	0.5037	0.8148	0.3111
Square_{250}	Recon. (Node)	0.8931	0.9989	0.1058	0.4516	0.7894	0.3378
Circle_{50}	Forecast (Edge)	0.9994	0.9999	0.0005	0.8993	0.8863	-0.0130
Square_{50}	Forecast (Edge)	0.9990	0.9990	0.0000	0.8037	0.8411	0.0374
Circle_{250}	Forecast (Edge)	0.9981	0.9987	0.0006	0.8418	0.8148	-0.0270
Square_{250}	Forecast (Edge)	0.9988	0.9989	0.0001	0.7835	0.7894	0.0059

References

[1] A. Chatterjee and B. S. Ahmed, "IoT Anomaly Detection Methods and Applications: A Survey," *Internet of Things*, vol. 19, p. 100568, 2022.

[2] H.-T. Nguyen, Q.-D. Ngo, and V.-H. Le, "A Novel Graph-Based Approach for IoT Botnet Detection," *International Journal of Information Security*, vol. 19, no. 5, pp. 567–577, 2020.

[3] Y. Zheng and et al, "Correlation-Aware Spatial-Temporal Graph Learning for Multivariate Time-Series Anomaly Detection," *IEEE transactions on neural networks and learning systems*, vol. 35, 2023.

[4] M. Zhao and O. Fink, "Dyedgegat: Dynamic Edge via Graph Attention for Early Fault Detection in IIOT Systems," *IEEE Internet of Things Journal*, vol. 11, no. 13, pp. 22 950–22 965, 2024.

[5] M. Navarro and et al, "Towards Long-Term Multi-Hop WSN Deployments for Environmental Monitoring: An Experimental Network Evaluation," *Journal of Sensor and Actuator Networks*, vol. 3, 2014.

[6] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly Detection and Diagnosis from System Logs Through Deep Learning," in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, 2017, pp. 1285–1298.

[7] Y. Liu, H. Wang, X. Zheng, and L. Tian, "An Eficient Framework for Unsupervised Anomaly Detection over Edge-Assisted Internet of Things," *ACM Trans. Sensor Netw.*, vol. 66, no. 10, pp. 550–4859, 2023.

[8] D. S. Kalpana, R. Thenmozhi, and D. S. Kumar, "Self-Healing IoT Sensor Networks with Isolation Forest Algorithm for Autonomous Fault Detection and Recovery," in *International Conference on Automation and Computation (AUTOCOM)*. IEEE, 2024.

[9] Y. Liu, W. Zhang, Z. Bao, X. Chai, M. Gu, W. Jiang, Z. Zhang, Y. Tian, and F.-Y. Wang, "Brain-like Cognition-Driven Model Factory for IIoT Fault Diagnosis by Combining LLMs with Small Models," *IEEE Internet of Things Journal*, vol. 12, no. 16, 2024.

[10] L. Zhang, T. Jia, and et al, "ScalaLog: Scalable Log-Based Failure Diagnosis Using LLM," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2025.

[11] H. Guo, Z. Zhou, D. Zhao, and W. Gaaloul, "EGNN: Energy-Efficient Anomaly Detection for IoT Multivariate Time Series Data Using Graph Neural Network," *Future Generation Computer Systems*, vol. 151, no. C, pp. 45–56, 2024.

[12] T. K. K. Ho, A. Karami, and N. Armanfard, "Graph Anomaly Detection in Time Series: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 47, no. 8, pp. 6990–7009, 2025.

[13] H. Qiao, H. Tong, B. An, I. King, C. Aggarwal, and G. Pang, "Deep Graph Anomaly Detection: A Survey and New Perspectives," *IEEE Transactions on Knowledge and Data Engineering*, vol. 37, no. 9, pp. 5106–5126, 2025.

[14] A. Komadina, M. Martinić, S. Gro, and Mihajlović, "Comparing Threshold Selection Methods for Network Anomaly Detection," *IEEE Access*, vol. 12, pp. 124 943–124 973, 2024.

[15] B.-J. Min, J. Yoo, S. Kim, D. Shin, and D. Shin, "Network Anomaly Detection Using Memory-Augmented Deep Autoencoder," *IEEE Access*, vol. 9, pp. 104 695–104 706, 2021.