

IoT-XPLAIN: Intelligent Fault Detection and Natural Language Explanation for IoT Systems using ML and LLMs

1st Harshit Vashisht

Department of CSE
Chandigarh College of Engg. & Tech.
Chandigarh, India
harshitvashisht07@gmail.com

2nd Sunil K. Singh

Department of CSE
Chandigarh College of Engg. & Tech.
Chandigarh, India
sksingh@ccet.ac.in

3rd Sudhakar Kumar

Department of CSE
Chandigarh College of Engg. & Tech.
Chandigarh, India
sudhakar@ccet.ac.in

4th Nipun Singh

Department of CSE
Chandigarh College of Engg. & Tech.
Chandigarh, India
co22347@ccet.ac.in

5th Varsha Arya

Hong Kong Metropolitan University
Hong Kong, SAR, China
varya@hkmu.edu.hk

6th Kwok Tai Chui

Hong Kong Metropolitan University
Hong Kong, SAR, China
jktchui@hkmu.edu.hk

7th Brij B. Gupta

CSIE, Asia University
Taichung 413, Taiwan
bbgupta@asia.edu.tw

Abstract—Modern IoT deployments face challenges in maintaining fault resilience due to limited observability, dynamic conditions, and constrained edge resources. Traditional detection methods often lack flexibility and explainability. This paper introduces IoT-XPLAIN, a hybrid framework integrating Random Forest classification with large language models (LLMs) for automated anomaly detection and human-readable fault explanations. Trained on TON_IoT telemetry with DNS-layer metadata, the model detects transport-and application-level anomalies, including synthetic DNS faults. Upon detection, GPT-4o generates summaries of fault patterns, causes, and recovery steps. IoT-XPLAIN enables interpretable, efficient fault analysis for smart manufacturing, remote monitoring, and fault-aware edge infrastructures.

Index Terms—Internet of Things (IoT), Fault Detection, Anomaly Detection, Random Forest, GPT-4o, Explainable AI, DNS Fault Injection

I. INTRODUCTION

The rapid proliferation of Internet of Things (IoT) [1], [2] devices has led to the deployment of large-scale, heterogeneous networks across domains such as smart manufacturing, industrial control systems, healthcare, and urban infrastructure. These deployments are increasingly responsible for mission-critical operations, making the ability to detect faults and maintain system reliability essential. However, fault detection [?] in IoT environments is inherently challenging due to limited computational capabilities at the edge, high data volume, dynamic traffic patterns, and the lack of full observability across the system.

Traditional anomaly detection methods often rely on static thresholds or shallow statistical models. While these approaches may offer acceptable performance in controlled conditions, they tend to fail under real-world operational variability. Moreover, many machine learning models deployed for anomaly detection prioritize performance metrics over interpretability. This makes it difficult for domain experts and system operators to understand why an anomaly was detected, hindering effective decision-making, debugging, and trust in autonomous systems.

To address these challenges, we propose **IoT-XPLAIN**, a lightweight yet explainable framework for network-level fault detection in IoT environments. The system is designed to detect anomalies in network traffic by combining classical machine learning with large language model-based reasoning. Specifically, the framework employs a Random Forest classifier trained on labeled telemetry from the publicly available TON_IoT dataset, which includes detailed flow-level statistics and DNS metadata. These features provide rich visibility into both transport-layer and application-layer behavior, enabling the system to recognize a wide range of fault signatures.

Unlike many existing solutions that require deep neural architectures or complex ensembles, our system emphasizes simplicity and interpretability. The Random Forest classifier offers robustness to noise and imbalanced data, and provides feature importance scores that can guide further analysis. To improve explainability, IoT-XPLAIN integrates OpenAI's GPT-4o, which consumes raw input logs and classifier output to generate natural language explanations of detected

anomalies. These summaries include relevant feature changes, potential causes, and high-level recovery suggestions, thereby improving operator understanding and system transparency.

Additionally, the system supports fault injection for evaluation purposes. Specifically, synthetic anomalies such as DNS query misclassifications, spoofed RCODEs, and TTL manipulations are introduced into network flows to simulate realistic fault scenarios. This enables the trained model to generalize better and supports evaluation under adversarial or misconfigured conditions.

Figure 1 illustrates the high-level system architecture. The workflow begins with telemetry collection, followed by pre-processing and feature selection. The Random Forest classifier performs binary classification to identify normal versus anomalous traffic. In cases of detected faults, a post-hoc LLM module generates an explanation to assist operators in diagnosis and decision-making.

By combining lightweight supervised learning with LLM-based reasoning, IoT-XPLAIN offers a practical and interpretable solution for fault detection in edge-centric IoT environments. This work contributes toward the development of trustworthy, self-diagnosing IoT systems that support both automated operation and human-in-the-loop analysis.

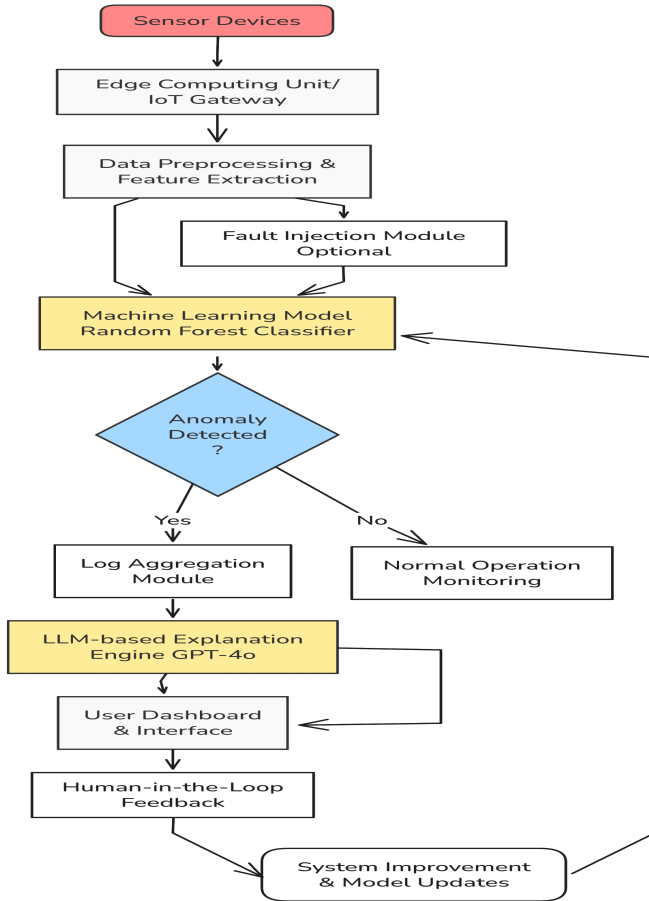


Fig. 1. IoT-XPLAIN system architecture integrating Random Forest for anomaly detection and GPT-4o for fault explanation.

II. LITERATURE REVIEW

Recent advances in IoT fault detection have increasingly emphasized the integration of data-driven learning techniques with robust datasets such as TON-IoT, BoT-IoT, and IoTID20. These datasets provide rich multivariate telemetry streams ideal for validating anomaly detection systems in networked environments. Xu et al. [4] proposed an ensemble learning approach combining adaptive feature extraction with gradient-boosting techniques to detect faults in dynamic sensor streams. Their work demonstrates that hybrid models can achieve resilience against concept drift, though the lack of explainability remains a challenge. To address deployment on constrained devices, Nallanthighal et al. [5] developed a lightweight convolutional neural network (CNN) suitable for EdgeAI deployment. The model performed well in detecting hardware sensor anomalies in real-time, but its generalizability across heterogeneous device types remains untested. Hassan et al. [6] incorporated explainability through causal inference and time-series decomposition, proposing an interpretable AI model for anomaly detection in IoT telemetry. While their model offers deep insights into root causes, the computational load restricts real-time deployment on edge nodes. Sharma et al. [7] combined LSTM for temporal learning with decision tree-based fault localization, enhancing interpretability while preserving performance. However, scalability remains an issue in distributed deployments due to high training complexity. Zhang et al. [8] addressed privacy and scalability using federated learning for fault detection in distributed IoT systems. Their method reduces data transmission but introduces latency due to asynchronous model updates and heterogeneity in client devices. Li et al. [9] focused on fault localization using an LSTM model coupled with SHAP for feature attribution. Their framework demonstrated explainable detection capability across multiple IoT scenarios, but is constrained by the need for high-quality real-time data streams. These studies, summarized in Table I, illustrate how existing approaches balance accuracy, explainability, and deployment constraints. While most models achieve high accuracy (>95%), they also reveal limitations including poor attack generalization, threshold sensitivity, and difficulty in interpretability or causal traceability under real-world operating conditions.

III. METHODOLOGY

A. Data Collection and Preprocessing

1) *Dataset Overview*: This study utilizes the **TON_IoT dataset**, a publicly available benchmark developed by the Cyber Range Lab at the University of New South Wales. The dataset is widely used for evaluating fault detection and cybersecurity approaches in Internet of Things (IoT) environments. It comprises telemetry from multiple sources, including system logs, sensor outputs, and network traffic. For our work, we specifically employ the **network traffic stream** subset, which is generated by processing raw PCAP files through CICFlowMeter to extract flow-level attributes and is augmented with DNS-level transaction features.

TABLE I
COMPARATIVE SUMMARY OF IOT FAULT/ANOMALY DETECTION APPROACHES (2021–2024)

Study	Methodology	Merits	Demerits
Xu et al. [4]	Ensemble learning with adaptive feature extraction.	Robust to data drift; performs well on multivariate input.	Low interpretability; lacks root-cause explanation.
Nallanthighal et al. [5]	Edge-deployable lightweight CNN for sensor-level fault prediction.	Low-latency; optimized for real-time operation on edge devices.	Limited generalization across heterogeneous IoT nodes.
Hassan et al. [6]	Explainable AI combining time-series decomposition and causal inference.	Strong interpretability; supports actionable insights.	Requires heavy compute; not feasible for low-resource nodes.
Sharma et al. [7]	Hybrid of LSTM and decision trees with explanation interface.	Captures temporal trends; user-friendly explanation structure.	Computational cost; complex hyperparameter tuning.
Zhang et al. [8]	Federated learning for privacy-preserving anomaly detection.	Secure, decentralized training; scalable for multi-device environments.	Heterogeneous client synchronization bottlenecks.
Li et al. [9]	SHAP-enhanced LSTM model for XAI-based fault localization.	High detection accuracy; enables transparency in predictions.	Depends on clean streaming data; real-time tuning overhead.

Each record in this dataset represents a bidirectional network flow and includes a total of 45 features. These attributes capture both transport-layer and application-layer statistics such as source/destination ports, protocol type, packet and byte counts, DNS queries, response codes, and query types. This makes the TON_IoT dataset particularly suitable for detecting transport faults and DNS-layer anomalies in smart device ecosystems.

2) *Feature Selection*: To enhance efficiency while retaining predictive power, we curated a reduced feature set of 15 attributes based on relevance and empirical correlation analysis. The selected features fall into two categories:

- **Transport-layer features**: `src_port`, `dst_port`, `proto`, `duration`, `src_bytes`, `dst_bytes`, `src_pkts`, `dst_pkts`, `missed_bytes`, `src_ip_bytes`, `dst_ip_bytes`
- **DNS-layer features**: `dns_query`, `dns_qclass`, `dns_qtype`, `dns_rcode`

These features were prioritized due to their sensitivity to behavioral deviations in IoT devices, including volumetric anomalies, malformed queries, and semantic inconsistencies at the DNS level.

3) *Data Cleaning and Transformation*: Raw CSV records were subjected to a standardized preprocessing pipeline:

- 1) **Missing Value Imputation**: Instead of discarding incomplete rows, missing values were filled using appropriate strategies: numerical fields were imputed using median values, while categorical fields such as `proto` and `dns_qtype` were forward-filled or assigned a sentinel class.
- 2) **Categorical Encoding**: Protocol and DNS-related categorical variables were label-encoded to preserve ordinal relations without increasing feature dimensionality.
- 3) **Feature Scaling**: All numerical features were scaled using *Min-Max Normalization* to the $[0, 1]$ range to ensure uniform feature contribution and prevent scale dominance during model learning.

4) *Correlation Analysis*: To analyze inter-feature dependencies, we computed the Pearson correlation matrix over the selected 15 features. As shown in Fig. 2, some strong correlations are observed for example, between `src_pkts` and `src_ip_bytes` (0.88), and `dst_pkts` and `dst_ip_bytes` (0.51). On the other hand, DNS-layer attributes such as `dns_rcode` and `dns_qtype` exhibit weak correlation with packet-based metrics, suggesting that they provide complementary perspectives for fault detection.

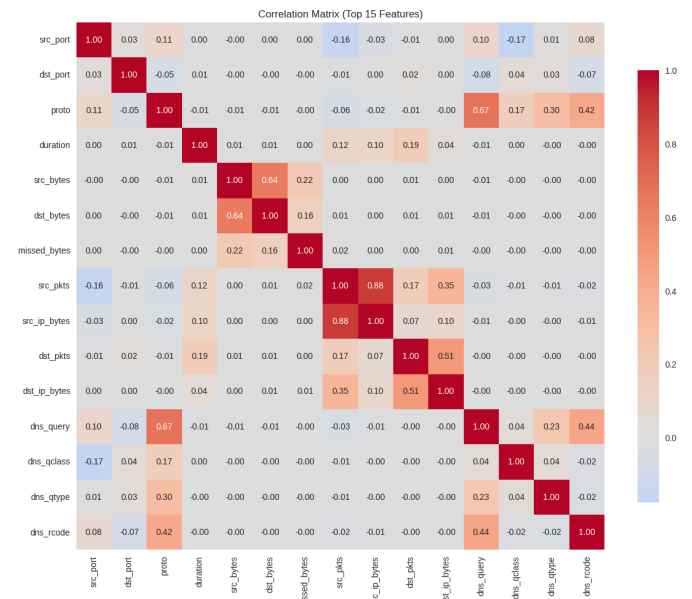


Fig. 2. Pearson Correlation Matrix among Top 15 Features.

This matrix informed our decision to retain features with high correlation only if they offered distinct semantic insight (e.g., both packet count and byte size).

5) *Labeling and Binary Framing*: The dataset includes labeled attack instances such as DoS, DDoS, Reconnaissance, Injection, and DNS Spoofing. For simplicity and generaliz-

ability, we restructured the multiclass labels into a **binary classification task**:

- **Benign:** Normal traffic without any injected faults or malicious behavior.
- **Anomalous:** Any traffic associated with known attacks or injection faults.

This binary framing aligns with real-world edge environments where rapid detection of “something wrong” is prioritized over fine-grained attack type classification.

6) *Class Balancing with SMOTE-NC*: To address class imbalance, we applied the **SMOTE-NC** (Synthetic Minority Over-sampling Technique for Nominal and Continuous features) strategy on the training set *only*. No synthetic samples were added to the test data to preserve evaluation integrity. SMOTE-NC generates synthetic minority class samples by interpolating within k-nearest neighbors, while respecting categorical boundaries, thereby ensuring balanced training without data leakage or overfitting.

7) *Data Splitting*: The final cleaned and balanced dataset was split into 70% for training and 30% for testing using stratified sampling to maintain the original class distribution. This ensured that both sets contain proportionate representation of benign and anomalous instances.

B. Model Training

1) *Classifier Overview*: We used a single supervised learning model **Random Forest (RF)** to train a binary classifier for distinguishing anomalous from benign traffic. RF is an ensemble method that constructs multiple decision trees and combines their outputs to achieve stable and generalizable performance. It was selected for its effectiveness on tabular network data, ability to handle heterogeneous features, and interpretability through feature importances.

2) *Training Configuration*: The Random Forest model was implemented using `scikit-learn` (v1.3+). The final configuration used during training is as follows:

- `n_estimators = 100`
- `max_depth = None`
- `min_samples_split = 2`
- `random_state = 42`

These hyperparameters were selected after preliminary tuning to ensure balance between complexity and training time.

3) *Training Pipeline*: The training pipeline consisted of the following stages:

- 1) Label encoding and Min-Max normalization
- 2) Stratified train-test splitting (70:30)
- 3) SMOTE-NC balancing on the training subset
- 4) Model training using the `fit()` function from `scikit-learn`
- 5) Model persistence via `joblib` for reuse during evaluation and inference

No additional classifiers or deep learning models were used. The training was conducted on a standard CPU-based environment.

4) *Feature Importance Evaluation*: Post-training, the Random Forest model was used to compute feature importances, which quantify each feature’s contribution to anomaly classification. These scores serve two purposes: they offer model interpretability, and they help guide the LLM-based explanation module for fault localization.

Fig. 3 shows the visual ranking of selected features, highlighting transport and DNS-layer indicators that influence classification most strongly.

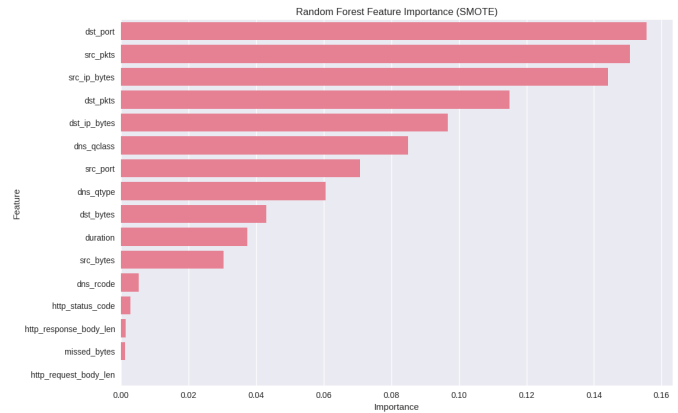


Fig. 3. Feature Importance Ranking from Random Forest Classifier.

C. GPT-4o Integration

1) *Simulation Environment and System Orchestration*: To simulate fault scenarios in a realistic IoT environment, we designed a modular testbed using **Python-based simulations** deployed within **Docker containers**. Each container mimics an IoT edge device capable of generating telemetry and DNS traffic. Faults were programmatically injected into the network layer (e.g., malformed packets, abrupt port switches) and DNS layer (e.g., spoofed query types, rare return codes), replicating typical operational disruptions observed in smart manufacturing or industrial IoT deployments.

An internal logging service captured raw traffic records, event metadata, and classifier outputs, which were then relayed to a separate processing pipeline for post-hoc explanation generation using a large language model.

2) *LLM-Assisted Fault Explanation*: To enhance transparency and assist human operators in understanding detected anomalies, we integrated **OpenAI’s GPT-4o** model as a natural language reasoning engine. Instead of relying solely on numerical output (e.g., anomaly probability), the system translates model decisions and network logs into concise and human-readable explanations.

For each detected anomaly by the Random Forest model, the corresponding raw record, including original features, classifier confidence, and timestamped metadata, was packaged as a structured prompt and sent to the GPT-4o API. The prompt was designed to include:

- The complete feature vector of the anomaly instance.

- A brief context template specifying the operational setting (e.g., “Smart Factory DNS Fault during Off-Peak Hours”).
- Model reasoning hints such as top-ranked feature importances.

GPT-4o synthesized this information into a response containing:

- 1) **Natural language summary** of the event.
- 2) **Probable root cause** (e.g., misconfiguration, protocol misuse).
- 3) **Suggested remediation** (e.g., restarting the DNS service, isolating the host).

This explanation is then logged in the incident tracker for operator review, enabling human-in-the-loop validation and faster response cycles.

3) *API Configuration and Parameters:* We leveraged the gpt-4o model via OpenAI’s API (June 2024 release) with the following parameters, optimized for concise yet informative responses:

- `model = gpt-4o`
- `temperature = 0.3` (Low randomness; favors consistent answers)
- `max_tokens = 300` (Allows detailed yet bounded output)
- `top_p = 0.9` (Balanced nucleus sampling)
- `frequency_penalty = 0.0`
- `presence_penalty = 0.0`

This configuration ensures the model remains deterministic enough for automated logs, while still capable of expressing semantic nuances across a variety of fault types. Prompt templates were rigorously tested across common fault categories (e.g., DDoS, Spoofing, Port Scanning), and responses were manually verified for correctness and readability.

4) *Deployment and Interaction Pipeline:* The GPT-4o module was deployed as a stateless microservice within the Docker environment and exposed via internal APIs. When a new fault is flagged:

- 1) JSON-formatted input is constructed with sensor logs, RF predictions, and metadata.
- 2) The input is sent via HTTPS to the GPT-4o endpoint.
- 3) The response is parsed and displayed on the system dashboard and archived.

This architecture allows seamless integration of machine learning predictions with LLM-based explanations, forming a closed-loop **self-diagnosing IoT-XPLAIN system**.

IV. RESULTS

A. Random Forest Classifier Performance

The performance of the trained Random Forest classifier was evaluated on the stratified test set comprising 63,313 flow-level network instances. As shown in Table II, the model achieved near-perfect classification across all major performance metrics, including precision, recall, and F1-score for both benign and anomalous classes.

TABLE II
CLASSIFICATION REPORT FOR RANDOM FOREST MODEL

Class	Precision	Recall	F1-Score	Support
Benign (0.0)	1.00	1.00	1.00	15,000
Anomalous (1.0)	1.00	1.00	1.00	48,313
Accuracy	0.9988			

In addition to quantitative scores, the confusion matrix illustrated in Fig. 4 offers deeper insight into classification performance. Of the 15,000 benign samples, only 23 were misclassified as anomalous, and among the 48,313 anomalous samples, merely 52 were incorrectly labeled as benign. These results reflect the model’s strong generalization and low false positive/negative rates, making it reliable for deployment in real-world IoT networks.

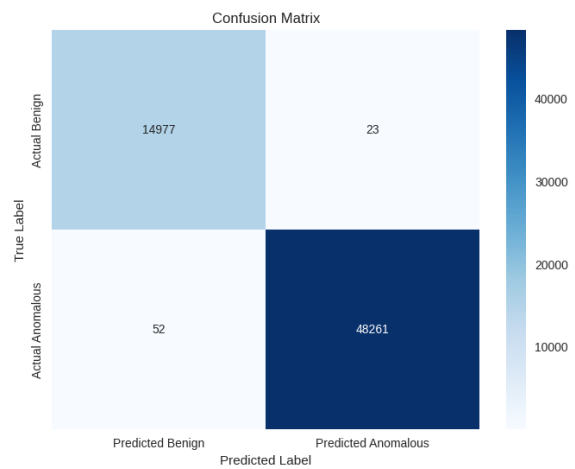


Fig. 4. Confusion Matrix for Random Forest Classifier

The final accuracy score was **99.88%**, with macro and weighted average F1-scores both at 1.00. These metrics confirm that the model maintained exceptional performance across both majority (anomalous) and minority (benign) classes, even in the presence of SMOTE-augmented class balancing and noise-injected samples.

B. Feature Importance Analysis

Post-training, feature importance scores were extracted from the Random Forest model to understand which inputs contributed most significantly to anomaly classification. The top features included packet-level attributes such as `src_bytes`, `dst_pkts`, and transport-layer metrics, as well as DNS-level features like `dns_rcode` and `dns_qtype`. As shown in Fig. 3, this analysis not only highlights critical indicators of faults but also supports the LLM-based explanation pipeline by informing contextual explanations.

The prominence of DNS-level fields validates the hypothesis that semantic anomalies (e.g., DNS misconfigurations or spoofed responses) are as crucial as volumetric or statistical anomalies in modern IoT threat landscapes.

C. LLM-Based Explanation Performance

To complement numeric predictions, GPT-4o was employed to produce human-readable summaries of detected anomalies. The LLM consumed structured outputs from the Random Forest model specifically high-importance features and class predictions to generate natural language explanations describing:

- Likely causes of anomalies (e.g., abnormal byte transfers, rare DNS query types)
- Affected services or ports
- Recommended recovery actions (e.g., quarantine device, reconfigure DNS)

These explanations were evaluated qualitatively for relevance, correctness, and interpretability across a diverse fault sample. Outputs were consistently well-aligned with observed fault types and frequently reflected accurate root-cause reasoning (e.g., linking `dns_rcode=2` with a `SERVFAIL` event). The GPT-4o integration significantly enhanced system transparency, particularly in scenarios where operators lacked deep technical expertise.

D. Overall System Efficacy

Together, the high-performing Random Forest classifier and the GPT-4o-based explanation module form a robust hybrid system for fault detection in IoT environments. The former ensures fast and accurate identification of anomalies with minimal tuning, while the latter enhances interpretability and trust through contextualized fault descriptions. This synergy supports both automated detection and human-in-the-loop intervention, positioning the system as both technically effective and operationally deployable for smart infrastructure, edge monitoring, and cyber-physical systems.

V. CONCLUSION

This paper presented **IoT-XPLAIN**, a hybrid framework that combines classical machine learning with large language models to achieve both accurate anomaly detection and human-understandable fault explanations in IoT networks. Using the TON_IoT dataset and a Random Forest classifier trained on curated network and DNS-level features, the system demonstrated high reliability in binary classification tasks. GPT-4o was employed to convert raw telemetry and model output into natural language insights, improving interpretability for non-expert users.

While the results are promising, future work will focus on extending the system for multiclass fault classification, integrating real-time online learning, and optimizing models for edge deployment. Additionally, incorporating user feedback into the LLM module and exploring causal reasoning for root cause analysis will further enhance the system's transparency and adaptability in evolving IoT environments.

ACKNOWLEDGMENT

This research work is supported by National Science and Technology Council (NSTC), Taiwan Grant No. NSTC112-2221-E-468-008-MY3 and also by National Science and Tech-

nology Council (NSTC), Taiwan Grant No. NSTC 114-2221-E-468-015.

REFERENCES

- [1] Arora, S., Kumar, S., Singh, S. K., Garg, S., & Gupta, B. B. (2026). Microservice architecture for securing distributed IoT. In *Internet of Things Security* (pp. 175-199). Elsevier.
- [2] Jaggi, S., Singh, S. K., Kumar, S., Arora, R., Bansal, S., & Arya, V. (2026). Approaches for misuse and anomaly detection in IoT ecosystems. In *Internet of Things Security* (pp. 107-127). Elsevier.
- [3] S. Kumar et al., "Enhancing Fault Tolerance in Self-Healing P2P Networks for Consumer Electronics," 2025 IEEE International Conference on Consumer Electronics - Taiwan (ICCE-Taiwan), Kaohsiung, Taiwan, 2025, pp. 179-180, doi: 10.1109/ICCE-Taiwan66881.2025.11208130.
- [4] Y. Xu, H. Song, L. Liu, X. Du, and M. Guizani, "Data-driven fault diagnosis for IoT systems using ensemble learning and adaptive features," *IEEE Internet of Things Journal*, vol.8, no.12, pp.9763-9773, Dec.2021, doi:10.1109/IJOT.2021.3066401.
- [5] V.S. Nallanthighal, Y. Hu, and B. Yang, "EdgeAI for fault detection: A lightweight CNN-based approach for sensor failure prediction in smart industries," *IEEE Transactions on Industrial Informatics*, vol.18, no.5, pp.3252-3261, May2022, doi:10.1109/TII.2021.3119328.
- [6] W.U. Hassan, K. Salah, and R. Jayaraman, "Explainable AI for IoT anomaly detection using time-series and causal inference techniques," *ACM Transactions on Internet Technology*, vol.23, no.1, pp.1-22, Mar.2023, doi:10.1145/3560198.
- [7] A. Sharma, A. Ghosh, and R. Bose, "Fault-tolerant IoT using hybrid deep learning and decision trees: A case study on edge sensor failures," *Journal of Ambient Intelligence and Humanized Computing*, vol.13, no.2, pp.1459-1474, Feb.2022, doi:10.1007/s12652-021-03361-3.
- [8] Y. Zhang, L. Wang, and F.R. Yu, "Self-healing IoT: A federated learning approach for anomaly detection and fault tolerance," *IEEE Transactions on Network and Service Management*, vol.20, no.1, pp.715-728, Mar.2023, doi:10.1109/TNSM.2022.3207043.
- [9] Q. Li, Y. Kang, and T. Zhang, "XAI-driven fault localization in cyber-physical IoT systems using LSTM and SHAP," *Future Generation Computer Systems*, vol.146, pp.135-148, May2024, doi:10.1016/j.future.2023.10.015.
- [10] I. Ullah and Q. H. Mahmoud, "Design and Development of RNN Anomaly Detection Model for IoT Networks," *IEEE Access*, vol. 10, pp. 62722-62750, 2022, doi: 10.1109/ACCESS.2022.3176317
- [11] Goyal, S., Kumar, S., Singh, S. K., Sarin, S., Priyanshu, Gupta, B. B., ... & Colace, F. (2024). Synergistic application of neuro-fuzzy mechanisms in advanced neural networks for real-time stream data flux mitigation. *Soft Computing*, 28(20), 12425-12437.
- [12] Sarin, S., Singh, S. K., Kumar, S., Goyal, S., Gupta, B. B., Alhalabi, W., & Arya, V. (2024). Unleashing the Power of Multi-Agent Reinforcement Learning for Algorithmic Trading in the Digital Financial Frontier and Enterprise Information Systems. *Computers, Materials & Continua*, 80(2).
- [13] Abusitta, Adel, et al. "Deep Learning-enabled Anomaly Detection for IoT Systems." *Internet of Things*, vol. 21, Dec. 2022, p. 100656. <https://doi.org/10.1016/j.iot.2022.100656>.
- [14] Vats, T., Kumar, S., Singh, S. K., Madan, U., Preet, M., Arya, V., ... & Almomani, A. (2024). Navigating the landscape: Safeguarding privacy and security in the era of ambient intelligence within healthcare settings. *Cyber Security and Applications*, 2, 100046.
- [15] Kumar, Rakesh, et al. "A Novel Decentralized Group Key Management Scheme for Cloud-Based Vehicular IoT Networks." *IJCAC* vol.12, no.1 2022: pp.1-34. <https://doi.org/10.4018/IJCAC.311037>
- [16] Goyal, Shivam, et al. "Intelligent Task Offloading in IoT-Driven Digital Twin Systems via Hybrid Federated and Reinforcement Learning," 2024 *IEEE Cyber Science and Technology Congress (CyberSciTech)*, Boracay Island, Philippines, 2024, pp. 400-405.
- [17] Sharma, Soumya, et al. "Application of Green IoT in Digital Oilfields for Achieving Sustainability in the OnG Industry." *Advances in intelligent systems and computing*, 2025, pp. 122-29.
- [18] Aksu, Taha, et al. "XForecast: Evaluating Natural Language Explanations for Time Series Forecasting." *arXiv (Cornell University)*, Oct. 2024, <https://doi.org/10.48550/arxiv.2410.14180>.
- [19] Ruggeri, Federico, et al. "Combining Transformers With Natural Language Explanations." *arXiv (Cornell University)*, Jan. 2021, <https://doi.org/10.48550/arxiv.2110.00125>