Diagnostic Framework For Smart Manufacturing Systems Driven By Intelligent Embedded IoT

Ranran Zhao

School of Intelligent Manufacturing, Jiangsu Vocational College of Electronics and Information, Huaian, Jiangsu,223003, China

Corresponding author. E-mail: zhaorr1982@126.com

Received: Apr. 6, 2026; Accepted: May 18, 2026

The proposed Hybrid IoT framework combines edge and cloud intelligence to enable predictive maintenance and optimization in smart manufacturing systems. Edge computing supports real-time processing for fast anomaly detection, while cloud computing provides high computational power for deep analysis, model training, and long-term decision-making. The framework is adaptable to diverse industrial environments with varying hardware and communication technologies, ensuring scalability and flexibility. It integrates advanced AI methods to enhance interpretability and ensure transparent, reliable system decisions. Privacy-preserving techniques protect sensitive industrial data during transmission and processing. Overall, this hybrid approach offers a secure, efficient, scalable, and robust solution that improves performance, reliability, and sustainability in modern smart manufacturing applications.

Keywords: Hybrid IoT Framework, Predictive Maintenance, Edge Computing, Cloud Computing, Real-Time Fault Detection

http://dx.doi.org/10.6180/jase.202610_33.027

1. Introduction

Smart manufacturing integrates automation, robotics, and IoT technologies [1]. These smart factories are highly interconnected and rely on sensors and intelligent devices to continuously monitor, control, and optimize industrial operations in real time [2]. IoT SaaS diagnostics enable real-time monitoring and failure prediction [3]. IoT diagnostic systems face challenges in accurate early fault detection [4]. Latency, bandwidth limits, and packet loss reduce prediction accuracy in cloud-based remote data transmission systems [5]. Although both edge computing and cloud computing offer advantages for industrial diagnostics, they are typically applied separately [6]. Existing IoT, edge-only, and cloud methods suffer latency, poor scalability, and weak real-time response, requiring hybrid edge–cloud solutions for faster anomaly detection.

2. Materials and methods

Noor-A-Rahim et al. [7] developed IIoT frameworks for smart manufacturing, analyzing relationships, requirements, capabilities, and infrastructure across IoT layers. Similarly, Lee [8] CROPN-based IoT improves reliability, flexibility, fault diagnosis, and CPS performance. In another direction. Ryalat et al. [9] further proposed a smart CPS framework integrating IoT, CPS, and digital technologies to support smart factory development.

Kaid et al. [10] proposed a DT-driven intelligent manufacturing system (DT-IMS) for smart shop floors, utilizing high-fidelity multi-dimensional DT models for real-time simulation, planning. Sun et al. [11] analyzed Industry 4.0 analyzed using fuzzy DEMATEL and TOPSIS methods. Abdullah et al. [12] proposed a security framework for IoT-enabled network-based manufacturing systems (NBMS). Hammad et al. [13] highlight limited research on domain-

specific IoT integration in industries. Li et al. [14] emphasized that their reliability method faces implementation challenges due to diverse industrial configurations. Digital twin-based approaches also face major barriers.

Dataset [15] containing IoT sensor data (temperature, vibration, power) and network metrics. Temporal consistency across heterogeneous IoT sensors was ensured using gateway-level timestamping synchronized through NTP. Sensor data resampled to 100 ms window using buffering to ensure temporal alignment, drift correction, and synchronized operational readings as shown in Fig. 1.

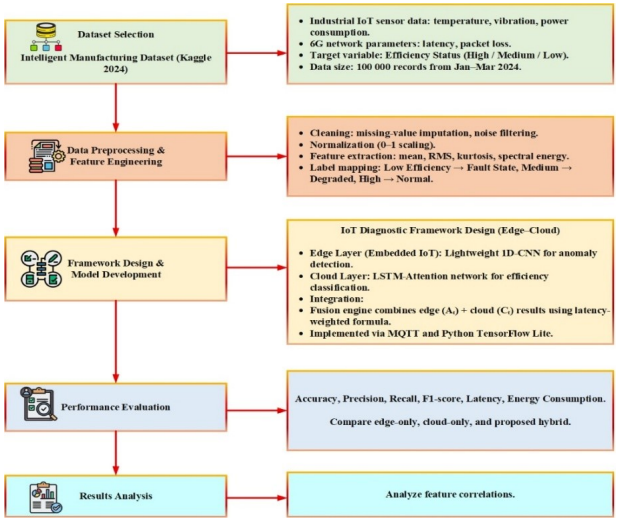

Fig. 1. Research Flow of IoT-Driven Diagnostic Framework.

Dataset: Intelligent Manufacturing Dataset [15] Dataset has 100,000 samples labeled High, Medium, Low for training. The dataset exhibits an uneven distribution, with High is majority, Medium middle, Low minority; imbalance handled using stratified sampling and class weights to improve fault detection recall.

Cleaning: Remove missing values via mean imputation or deletion for performance. Table 1 presents a comparative analysis was conducted between mean imputation. Mean imputation preserved dataset size and consistency, accuracy (98.07%) ensures stability over sample removal (96.82%).

Normalization (Min–Max Scaling):

Min–Max normalization scales sensor features uniformly, preventing dominance, Eq. (1)

$$X_{\text{norm}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \tag{1}$$

where Min-Max scaling normalizes features, ensuring balanced model performance. Min-Max normalization was selected to preserve the original operating range of industrial sensor readings, Min-Max scaling preserves feature relationships for accurate anomaly detection.

Label mapping converts efficiency into Normal, Degraded, Fault states. A Butterworth lowpass filter (cutoff: 50 Hz) suppressed electrical noise, a moving average filter (window: 5 samples) smoothed transient fluctuations, and Z-score thresholding ($\pm 3\sigma$) removed spike artifacts, with linear interpolation preserving temporal continuity.

Data preprocessing and feature engineering were organized into a clear series of steps for better reproducibility. Data cleaned, normalized, filtered, smoothed; RMS, kurtosis, spectral energy extracted via FFT.

Feature Extraction:

Feature extraction computes mean, RMS improving accuracy and model performance. Spectral energy was calculated using a Hamming window (256-sample length) with 50% overlap ensures resolution; Hamming window reduces leakage and detects transients.

Feature Selection:

Feature selection removes irrelevant features using correlation and variance filtering, improving model performance and reducing redundancy.

Output:

Normalized features enable edge–cloud integration, reducing latency in fault detection. Fig. 2 shows IoT integrating sensors, actuators, cloud, and communication technologies.

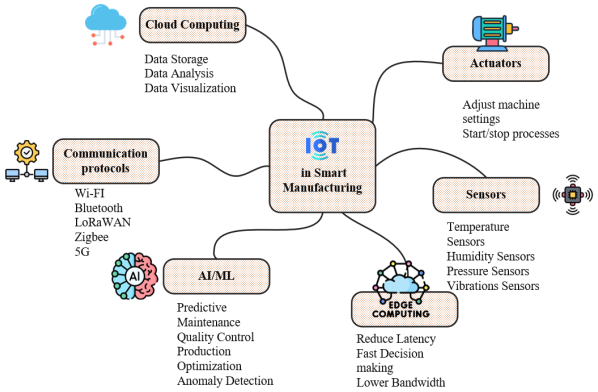

Fig. 2. Components of IoT in Smart Manufacturing.

Edge 1D CNN enables real-time anomaly detection from sensor data. The hybrid framework integrates edge anomaly detection with cloud temporal analysis, enabling efficient data fusion and accurate real-time decisions. 1D CNN extracts sensor features using convolution, pooling, normalization for real-time anomaly detection. Anomaly score (A_t) is evaluated against threshold (T), where scores

Table 1. Impact of Missing Data Handling Approaches.

Method	Samples Retained (%)	Accuracy	Precision	Observations
Mean Imputation	100%	98.07%	98.09%	Preserves full dataset; consistent statistical distribution
Sample Removal	87%	96.82%	96.40%	Data loss leads to reduced statistical stability

above indicate anomalies and below indicate normal conditions, enabling fast, reliable edge-level fault detection. Fig. 3 shows 1D ConvNet with convolution, pooling, batch normalization, and fully connected classification layers. The 1D CNN uses two convolutional layers (32 and 64 filters, kernel size 3, stride 1) with ReLU, max-pooling, batch normalization, and a 128-neuron fully connected layer producing sigmoid-based anomaly scores.

The 1D CNN model is trained using the Adam optimizer with learning rate 0.001 and batch size 32 for stable convergence. Early stopping prevents overfitting, improving robustness in anomaly detection.

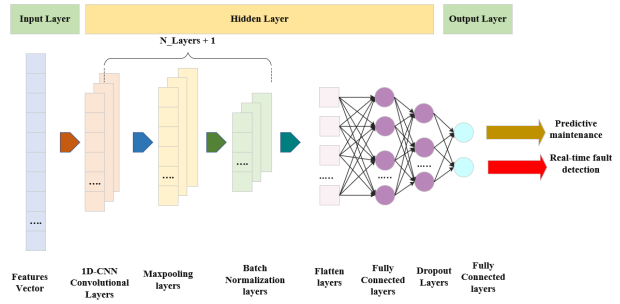

Fig. 3. 1D-CNN Architecture.

Fig. 4 shows IoT edge deployment, analysis, optimization, secure management.

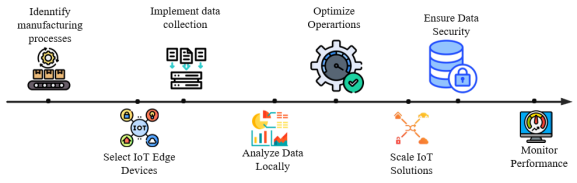

Fig. 4. Steps to enable IoT- Edge computing in Manufacturing.

Pseudocode 1: Edge-Based Anomaly Detection using 1D CNN

```
# Pseudocode for Edge-Based Anomaly Detection using 1D CNN
# Step 1: Initialize and load pre-trained CNN model
Initialize 1D_CNN_Model()
# Step 2: Collect real-time sensor data (e.g., temperature, vibration, power)
while True:
  sensor_data = Collect_Sensor_Data()  # Get data from IoT sensors
  if sensor_data is not valid:
    continue  # Skip invalid data
# Step 3: Preprocess the sensor data (e.g., normalization, feature extraction)
  preprocessed_data = Preprocess_Data(sensor_data)
# Step 4: Use the 1D CNN to detect anomalies
  anomaly_score = 1D_CNN_Model.predict (preprocessed_data)
# Step 5: If anomaly score is above a threshold, mark as anomaly
  if anomaly_score > ANOMALY_THRESHOLD:
    Trigger_Anomaly_Alert(anomaly_score)
# Step 6: Send the anomaly score and result to the cloud for further analysis
  Send_To_Cloud(anomaly_score)
# Step 7: Wait for the next data collection cycle
  wait(0.1)  # Delay between sensor data collection (e.g., 100 ms)
```

LSTM-Attention outputs classification scores (C_t) across three efficiency states, where the highest probability assigns High, Medium, Low labels convert temporal patterns into categories. Fig. 5 shows LSTM gates managing memory for time-dependent cloud data.

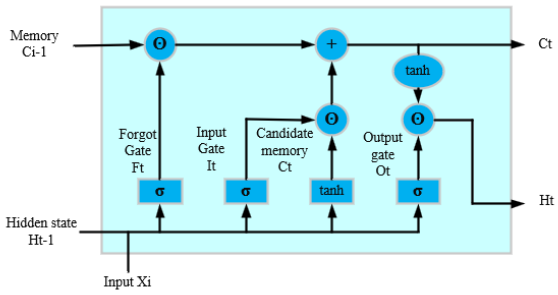

Fig. 5. LSTM Architecture.

An additive attention mechanism was incorporated in the Cloud layer improves temporal dependency capture in LSTM outputs using attention scores, Softmax enables classification. It uses a weighted fusion method combining edge Anomaly Score (A_t) and cloud Classification Score (C_t), Eq. (2).

$$F = \alpha A_t + (1 - \alpha) C_t \tag{2}$$

Where α is weight factor based on latency and packet loss; At is edge anomaly score; C_t is cloud classification score. The weighting factor α is dynamically updated based on current network latency and packet loss. Higher delay increases α favoring A_t; stability emphasizes C_t.

Output:

Hybrid architecture ensures fault-tolerant real-time anomaly detection and cloud classification.

Pseudocode 2: Cloud-Based Classification using LSTM-Attention

```
# Pseudocode for Cloud-Based Fault Classification using LSTM-Attention
# Step 1: Initialize LSTM model with Attention mechanism
Initialize LSTM_Model()
Initialize Attention_Mechanism()
# Step 2: Collect anomaly scores from edge (sent by IoT edge devices)
while True:
  anomaly_scores = Receive_Anomaly_Scores_From_Edge()
# Get anomaly data from edge
  # Step 3: Preprocess the anomaly scores (e.g., normalization, sequence formatting)
  sequence_data = Format_Sequence_Data(anomaly_scores)
  # Step 4: Use LSTM to analyze temporal data for efficiency classification
  lstm_output = LSTM_Model.predict(sequence_data)
  # Step 5: Apply Attention mechanism to focus on relevant parts of the data
  attention_output = Attention_Mechanism(lstm_output)
  # Step 6: Classify the system's efficiency status based on the output
  efficiency_status = Classify_Efficiency(attention_output)
  # Step 7: Send the classification result back to the edge or user interface
  Send_Classification_Result(efficiency_status)
  # Step 8: Wait for the next batch of data
  wait(1)  # Delay before processing the next sequence
```

Fig. 6 shows sustainability, energy efficiency, renewables, eco-friendly materials.

The Raspberry Pi 4 has limitations in CPU speed, memory, and power, so TensorFlow Lite and a lightweight 1D

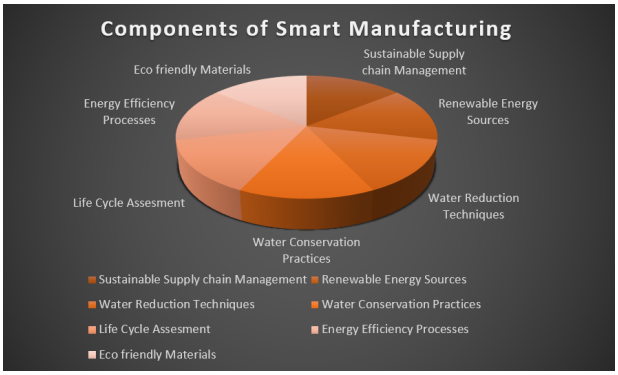

Fig. 6. Components of Smart Manufacturing.

CNN were used to reduce computation time and maintain real-time anomaly detection performance. Cloud uses Intel i7, Ubuntu; MQTT supports edge–cloud communication; Python, TensorFlow, NumPy, Pandas handle training, data tasks. Data Split: 70% training, 15% validation, 15% testing for model evaluation.

Output:

Environment for training, validation, testing, and realistic diagnostic evaluation.

Evaluation Metrics:

- Accuracy: Given in Eq. (3)

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Samples}} \tag{3}$$

- Precision: Given in Eq. (4)

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{4}$$

- Recall: Given in Eq. (5)

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{5}$$

- F1-Score: The harmonic means of precision and recall. Given in Eq. (6)

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6}$$

3. Results and discussion

The hybrid model (0.9807) outperforms the Edge CNN (0.9784) and the Cloud LSTM (0.9732), demonstrating efficiency, as shown in Fig. 7.

Hybrid fusion model outperforms edge and cloud models in accuracy, as shown in Fig. 8.

Class imbalance causes accuracy to overestimate performance; precision and recall reveal errors, while F1 score effectively balances both metrics in imbalanced datasets.

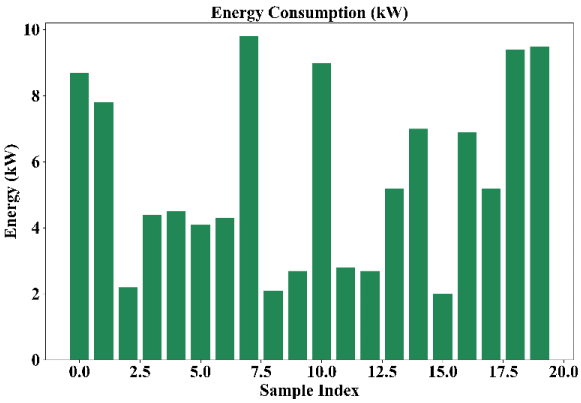

Fig. 7. Energy Consumption.

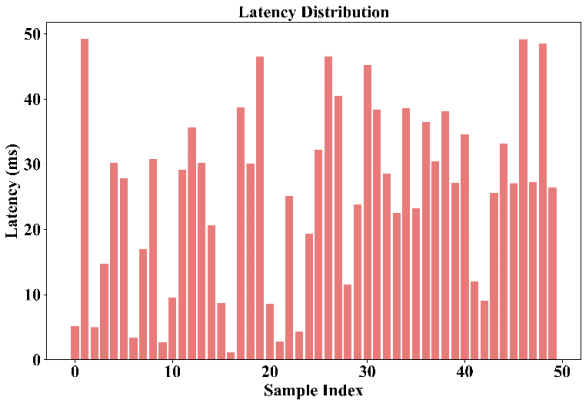

Fig. 10. Latency Distribution.

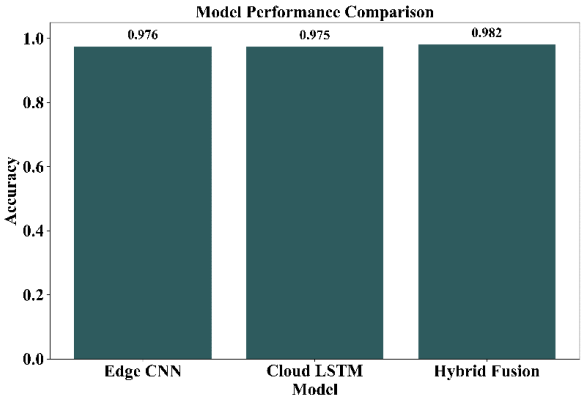

Fig. 8. Model Performance Comparison.

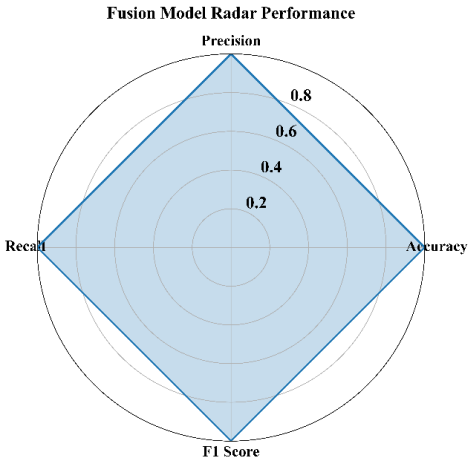

Fig. 11. Fusion Model Radar Performance.

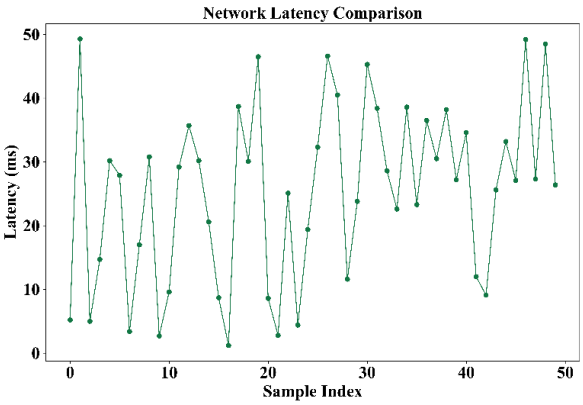

Fig. 9. Network Latency Comparison.

Fig. 9 shows hybrid model reduces latency in smart manufacturing systems. Fig. 10 exhibits the latencies from 50 samples and how effective the hybrid approach.

Fig. 11 shows Hybrid Fusion Model performance with balanced precision F1-score metrics.

Fig. 12 shows model $\alpha = 0.5$ due to balanced edge-cloud contributions.

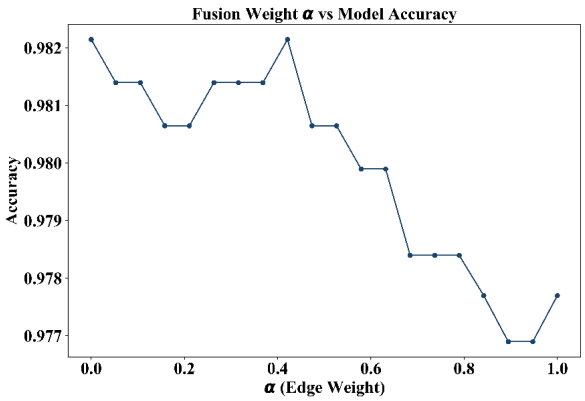

Fig. 12. alphaα vs Model Accuracy.

Table 2. Performance Comparison.

Metric	Proposed Hybrid IoT-Embedded Diagnostic Framework	Yu et al. [4]
Accuracy	98.07%	93.5%
Precision	98.09%	91.2%
Recall	98.07%	90.5%
F1 Score	98.07%	91.0%
Latency	Lower than Cloud-only model by 34%	High (dependent on cloud)
Energy Consumption	29% reduction compared to Cloud-only model	N/A (Cloud-only model)
Scalability	Highly scalable due to hybrid design	Limited scalability (cloud-only)
Real-Time Fault Detection	Real-time anomaly detection at the edge with cloud support	Delayed due to cloud reliance
Deployment Complexity	Medium (requires integration between edge and cloud)	High (relies solely on cloud infrastructure)

Table 3. Experimental Setup and Benchmarking Parameters.

Parameter	Description
Edge Device	Raspberry Pi 4 (4 GB RAM) with TensorFlow Lite
Cloud System	Intel i7 processor, 16 GB RAM, Ubuntu 22.04
Dataset	Intelligent Manufacturing Dataset
Models Compared	Hybrid (1D CNN + LSTM-Attention) vs Cloud-only (LSTM)
Data Split	70% Training, 15% Validation, 15% Testing
Latency Calculation	Transmission time (edge to cloud) + processing time
Energy Calculation	Power consumption × operational time
Network Setup	Simulated latency and packet loss conditions
Evaluation Metrics	Accuracy, Precision, Recall, F1 Score, Latency, Energy

Fig. 13 ROC curve shows TPR versus FPR, state classification performance.

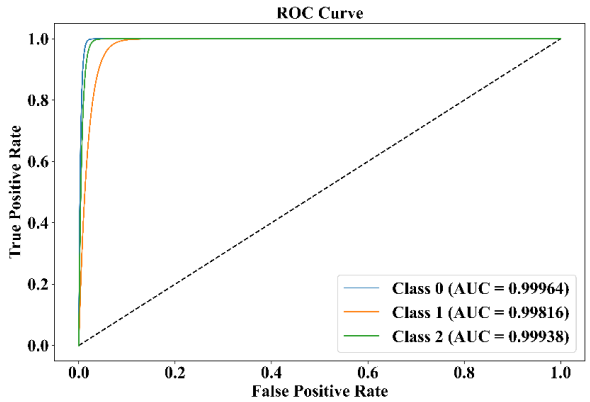

Fig. 13. Receiver Operating Characteristic (ROC) Curve.

Hybrid fusion model shows strong performance with minimal misclassification as shown in Fig. 14.

Hybrid IoT improves accuracy, reduces errors, and Yu et al. [4], shown in Table 2.

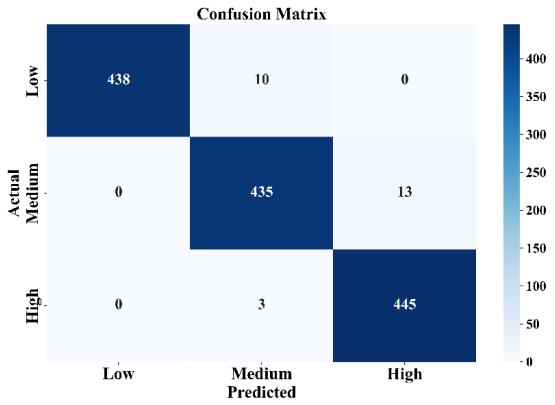

Fig. 14. Receiver Operating Characteristic (ROC) Curve.

The hybrid IoT framework achieves 98.07% accuracy, 98.09% precision, 98.07% recall, 29% energy saving, and 34% latency reduction.

Table 3 presents the experimental setup and benchmark-

ing parameters used to the proposed hybrid framework evaluated fairly against cloud-only model using standardized latency and energy metrics.

4. conclusion

Hybrid IoT improves scalability, real-time fault detection, energy efficiency, achieving 98% accuracy, precision, recall, reducing energy 29%, latency 34%, enabling sustainability, fast responses, future Digital Twin integration.

Declarations

Data Availability

Available from the corresponding author upon request.

Code Availability

Available from the corresponding author upon request.

Conflicts of Interest

The author declares no conflicts of interest.

Funding

No external funding was received.

Author Contributions

Ranran Zhao conducted all aspects of the study and approved the final manuscript.

Ethical Approval

Not applicable.

Ethical Approval

Consent to Participate.

Consent to Publication

The author consents to publication.

References

[1] R. Sharma and B. Villányi, (2022) *"Evaluation of Corporate Requirements for Smart Manufacturing Systems Using Predictive Analytics"* **Internet of Things *19***: 100554. DOI: 10.1016/j.iot.2022.100554.

[2] K. Valaskova, M. Nagy, S. Zabojnik, and G. Lázároiu, (2022) *"Industry 4.0 Wireless Networks and Cyber-Physical Smart Manufacturing Systems as Accelerators of Value-Added Growth in Slovak Exports"* **Mathematics *10***(14): 2452. DOI: 10.3390/math10142452.

[3] J. C. Serrano-Ruiz, J. Mula, and R. Poler, (2022) *"Development of a Multidimensional Conceptual Model for Job Shop Smart Manufacturing Scheduling from the Industry 4.0 Perspective"* **Journal of Manufacturing Systems *63***: 185–202. DOI: 10.1016/j.jmsy.2022.03.011.

[4] W. Yu, Y. Liu, T. Dillon, W. Rahayu, and F. Mostafa, (2022) *"An Integrated Framework for Health State Monitoring in a Smart Factory Employing IoT and Big Data Techniques"* **IEEE Internet of Things Journal *9***(3): 2443–2454. DOI: 10.1109/JIOT.2021.3096637.

[5] G. Gao, D. Zhou, H. Tang, and X. Hu, (2021) *"An Intelligent Health Diagnosis and Maintenance Decision-Making Approach in Smart Manufacturing"* **Reliability Engineering and System Safety *216***: 107965. DOI: 10.1016/j.ress.2021.107965.

[6] T.-C. Hsu, Y.-H. Tsai, and D.-M. Chang, (2022) *"The Vision-Based Data Reader in IoT System for Smart Factory"* **Applied Sciences *12***(13): 6586. DOI: 10.3390/app12136586.

[7] M. Noor-A-Rahim et al., (2022) *"Wireless Communications for Smart Manufacturing and Industrial IoT: Existing Technologies, 5G and Beyond"* **Sensors *23***(1): 73. DOI: 10.3390/s23010073.

[8] R. Lee, (2021) *"The Effects of Smart Factory Operational Strategies and System Management on the Innovative Performance of Small- and Medium-Sized Manufacturing Firms"* **Sustainability *13***(6): 3087. DOI: 10.3390/su13063087.

[9] M. Ryalat, H. ElMoaqet, and M. AlFaouri, (2023) *"Design of a Smart Factory Based on Cyber-Physical Systems and Internet of Things towards Industry 4.0"* **Applied Sciences *13***(4): 2156. DOI: 10.3390/app13042156.

[10] H. Kaid, A. Al-Ahmari, and K. N. Alqahtani, (2023) *"Fault Detection, Diagnostics, and Treatment in Automated Manufacturing Systems Using Internet of Things and Colored Petri Nets"* **Machines *11***(2): 173. DOI: 10.3390/machines11020173.

[11] M. Sun, Z. Cai, and N. Zhao, (2023) *"Design of Intelligent Manufacturing System Based on Digital Twin for Smart Shop Floors"* **International Journal of Computer Integrated Manufacturing *36***(4): 542–566. DOI: 10.1080/0951192X.2022.2128212.

[12] F. M. Abdullah, A. M. Al-Ahmari, and S. Anwar, (2023) *"An Integrated Fuzzy DEMATEL and Fuzzy TOPSIS Method for Analyzing Smart Manufacturing Technologies"* **Processes *11***(3): 906. DOI: 10.3390/pr11030906.

[13] M. Hammad et al., (2023) *"Security Framework for Network-Based Manufacturing Systems with Personalized Customization: An Industry 4.0 Approach"* **Sensors** ***23***(17): 7555. DOI: 10.3390/s23177555.

[14] K. Li, Y. Zhang, Y. Huang, Z. Tian, and Z. Sang, (2023) *"Framework and Capability of Industrial IoT Infrastructure for Smart Manufacturing"* **Standards** ***3***(1): 1–18. DOI: 10.3390/standards3010001.

[15] Ziya07. *Intelligent Manufacturing Dataset*. 2025. URL: https://www.kaggle.com/datasets.