International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 02 | Feb – 2025 DOI: 10.55041/ISJEM02267
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

An Intelligent Fault Detection Approach Based on Machine Learning in Wireless Sensor Networks

Jangam Bhargavi[1], Gonepally Adithya Kumar[2], Pusala Arun Kumar[3], Singarapu Charan Teja[4]
Assistant Professor of Department of CSE(AI&ML) of ACE Engineering College [1] Students of Department CSE(AI&ML) of ACE Engineering College [2,3,4]

Abstract
Wireless Sensor Networks (WSNs) play a crucial role in applications such as environmental monitoring, healthcare, and industrial automation. However, faults in sensor nodes due to energy depletion, hardware failures, or communication errors impact network efficiency and reliability. This paper surveys various approaches to fault detection in WSNs, emphasizing machine learning techniques such as K-Nearest Neighbors (KNN), Logistic Regression, and Random Forest. These methods enhance fault detection accuracy by analyzing sensor data patterns and classifying faulty nodes in real time. We also discuss existing techniques, their limitations, and the advantages of integrating machine learning for adaptive fault detection. Our analysis highlights the need for hybrid approaches that combine multiple algorithms to improve detection precision and minimize false positives.

Keyword: Fault Detection, Wireless Sensor Networks, Machine Learning, KNN Classifier, Logistic Regression, Random Forest.

1. Introduction
Wireless Sensor Networks (WSNs) are widely used for real-time data collection in various fields, including environmental monitoring, healthcare, and industrial automation. These networks consist of distributed sensor nodes that collect and transmit data wirelessly. However, due to factors such as hardware failures, environmental interferences, and energy depletion, sensor nodes may produce faulty readings or become inoperative. Detecting such faults in real time is critical to maintaining data integrity and ensuring network efficiency.

Wireless Sensor Networks (WSNs) are widely used for real-time data collection in various fields, including environmental monitoring, healthcare, and industrial automation. These networks consist of distributed sensor nodes that collect and transmit data wirelessly. However, due to factors such as hardware failures, environmental interferences, and energy depletion, sensor nodes may produce faulty readings or become inoperative. Detecting such faults in real time is critical to maintaining data integrity and ensuring network efficiency.

This project provides a comprehensive survey of fault detection techniques in WSNs, with a particular focus on machine learning models such as K-Nearest Neighbors (KNN), Logistic Regression, and Random Forest. These models are evaluated in terms of their accuracy, efficiency, and adaptability in real-time fault detection scenarios. By exploring the strengths and limitations of each approach, this study aims to contribute to the development of more robust and reliable fault detection systems for WSNs.

2. Literature survey
1. Title: Location-Aided Routing (LAR) in Mobile Ad Hoc Networks
Author: Y.B. Ko, N.H. Vaidya, 2008
This project introduces a routing algorithm that minimizes unnecessary route discovery by leveraging location information. The approach reduces routing overhead and improves efficiency in dynamic networks. However, it relies on accurate location data, which may not always be available, limiting its practicality in certain environments.

2. Title: Energy-Efficient Target Coverage in Multi-Hop Wireless Sensor Networks
Author: S. Biswas, R. Das, 2019
This project proposes a multi-hop approach to balance energy consumption across sensor nodes, improving network lifetime. The project demonstrates reduced communication overhead and enhanced energy efficiency. However, increased computational complexity is a drawback, as managing multi-hop communication requires additional processing power.

3. Title: Effective Data Gathering and Energy-Efficient Communication in Wireless Sensor Networks Using UAVs
Author: S. Sotheara, K. Aso, N. Aomi, S. Shimamoto, 2021
This project explores the use of Unmanned Aerial Vehicles (UAVs) to optimize data collection in Wireless Sensor Networks. UAV-assisted communication reduces the burden on sensor nodes, improving network longevity. However, the approach requires precise UAV path planning and additional infrastructure, which can increase operational costs.

4. Title: Strategies for Clustering in Wireless Sensor Networks Using Classical, Optimization, and Machine Learning Techniques
Author: J. Amutha, S. Sharma, S.K. Sharma, 2022
This project compares classical clustering methods such as

LEACH and HEED with optimization-based and machine learning-driven clustering techniques. It highlights the advantages of adaptive clustering in dynamic environments. However, implementing ML-based clustering increases computational complexity, making real-time deployment challenging.

5. Title: Hybrid Fault Detection Approach for Wireless Sensor Networks
Author: L. Huang, T. Zhang, 2023
This project explores a hybrid fault detection system combining rule-based filtering with machine learning models. The method improves accuracy in identifying faulty sensor nodes while maintaining computational efficiency. However, continuous retraining of models is required to adapt to evolving network conditions.

6. Title: Deep Learning-Based Fault Identification in IoT Systems
Author: K. Das, S. Roy, 2022
This project utilizes Convolutional Neural Networks (CNNs) for automated fault detection in IoT-based Wireless Sensor Networks. The approach enhances fault identification accuracy while reducing manual intervention. However, the high computational cost of deep learning models remains a challenge for real-time deployment in resource-constrained environments.

3. Existing System

The existing fault detection systems in Wireless Sensor Networks (WSNs) primarily use traditional approaches such as threshold-based detection, statistical analysis, and redundancy-based verification. These methods operate by defining fixed thresholds for sensor readings, where any deviation beyond the set limits is considered a fault. Statistical models use probabilistic estimations to detect abnormal sensor behavior, while redundancy-based techniques compare data from multiple sensors to identify inconsistencies. Though these techniques provide basic fault detection, they are limited in handling dynamic network conditions, evolving sensor failures, and large-scale deployments. Moreover, they require manual configuration and lack the ability to adapt to varying environmental factors, making them inefficient for real-time fault detection.

3.1 Drawbacks of Existing System

1. High False Positives and False Negatives
The reliance on fixed thresholds often results in incorrect fault detection. Environmental variations or minor fluctuations in sensor readings can be falsely labeled as faults, while genuine failures may go undetected.

2. Lack of Adaptability and Learning Capabilities
Traditional methods do not adapt to changing network conditions, such as sensor aging, environmental interference, or new fault patterns. This limits their effectiveness in long-term deployments.

3. Scalability Issues in Large Networks
As the network size grows, manually tuning detection thresholds becomes impractical. The processing and communication overhead required for redundancy-based detection also increase, reducing network efficiency.

4. Energy Inefficiency Leading to Reduced Network Lifetime
Frequent redundant transmissions and computations for fault verification consume significant battery power in sensor nodes. Since WSNs typically operate in energy-constrained environments, this significantly shortens network lifespan.

5. Inability to Detect Complex or Hidden Faults
Conventional approaches struggle with intermittent faults, sensor drift, and subtle failures that do not produce clear threshold violations. They are also ineffective in distinguishing between environmental noise and genuine faults.
Due to these drawbacks, traditional fault detection methods in WSNs are insufficient for modern, large-scale, and energy-constrained applications. To overcome these challenges, machine learning-based fault detection provides a more adaptive, accurate, and scalable solution, leveraging data-driven techniques to improve fault classification and real-time anomaly detection.

4. Proposed System

The proposed fault detection system in Wireless Sensor Networks (WSNs) leverages machine learning algorithms to enhance accuracy, adaptability, and efficiency in identifying faulty sensor nodes. Unlike traditional threshold-based methods, this system dynamically learns from sensor data patterns and adapts to changing network conditions. The approach integrates K-Nearest Neighbors (KNN), Logistic Regression and Random Forest to classify sensor readings as normal or faulty based on historical and real-time data. By combining these models, the system improves fault detection accuracy, minimizes false positives, and reduces the need for manual threshold adjustments. Additionally, it optimizes energy consumption by reducing unnecessary transmissions and redundant computations. The system is designed to operate in three modes: base station, sender, and receiver, ensuring efficient fault detection, real-time monitoring and enhanced network resilience.

4.1 Algorithms

The proposed fault detection system in Wireless Sensor Networks (WSNs) employs three machine learning algorithms: K-Nearest Neighbors (KNN), Logistic Regression, and Random Forest. These models analyze sensor data to classify nodes as faulty or non-faulty, enhancing accuracy and adaptability.

1. K-Nearest Neighbors (KNN)
KNN is a distance-based classification algorithm that determines the class of a test sample based on the majority vote of its K-nearest neighbors in the feature space. The most

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 02 | Feb – 2025 DOI: 10.55041/ISJEM02267
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

commonly used distance metric in KNN is Euclidean distance, given by:

$$d(X,Y) = \sqrt{\sum_{i=1}^{n}(X_i - Y_i)^2}$$

Where:

- X,Y are two points in an nnn-dimensional space.
- Xi,Yi are the individual feature values of the points.
- d(X,Y) is the distance between the two points.

2. Logistic Regression

Logistic Regression is a probabilistic model used for binary classification, predicting whether a sensor node is faulty (Y=1Y = 1Y=1) or non-faulty (Y=0Y = 0Y=0). The prediction is based on the sigmoid function, which transforms input data into a probability score:

$$P(Y=1|X) = \frac{1}{1 + e^{-(\beta_0 + \sum_{i=1}^{n} \beta_i X_i)}}$$

Where:

- P(Y=1|X) is the probability of the sensor being faulty.
- Xi represents the sensor features (e.g., temperature, voltage, signal strength).
- β0 is the bias term.
- βi are the model coefficients (weights).

If P(Y=1|X)P(Y=1 | X)P(Y=1|X) is greater than a threshold (usually 0.5), the node is classified as faulty; otherwise, it is non-faulty.

3. Random Forest

Random Forest is an ensemble learning method that constructs multiple decision trees and averages their predictions to improve classification accuracy. The decision trees split the data at each node based on a feature that maximizes the Information Gain (IG) or minimizes Gini Impurity.

1. Gini Impurity (used for node splitting):

$$G = 1 - \sum_{i=1}^{C} P_i^2$$

Where:

- PiP_iPi is the probability of class iii.
- CCC is the total number of classes (faulty or non-faulty).

2. Information Gain (IG) (used in entropy-based splitting):

$$IG = H(parent) - \sum_{j} \frac{|S_j|}{|S|} H(S_j)$$

Where:

- H(S) is the entropy of a set SSS.
- |Sj|/|S| represents the proportion of samples in subset Sj.

Random Forest reduces overfitting by training each tree on a different subset of the data and aggregating predictions through majority voting.

4.2 Architecture

This architecture is structured around a Base Station (BS), which serves as the central node responsible for collecting and processing data from multiple sensor clusters. Wireless Sensor Networks (WSNs) often employ cluster-based communication to enhance scalability and energy efficiency. In this architecture, the network is divided into clusters, where each cluster comprises multiple sensor nodes and a designated Cluster Head (CH). The CH is responsible for aggregating data from sensor nodes and transmitting it to the BS, reducing redundant transmissions and optimizing energy consumption.

Data transmission follows a predefined path using original routes (red dashed lines), ensuring a structured communication hierarchy. However, to improve network fault tolerance and reliability, alternative routes (green dashed lines) are dynamically selected in case of node failure, congestion, or energy depletion. This adaptive routing mechanism ensures seamless data transmission and enhances network resilience.

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 02 | Feb – 2025 DOI: 10.55041/ISJEM02267
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

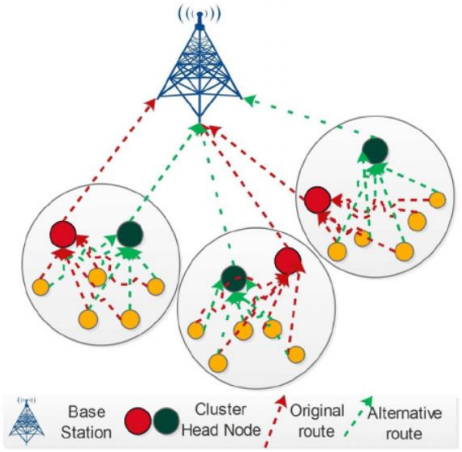

Figure: Architecture

The proposed model leverages hierarchical clustering techniques to prolong network lifespan, reduce communication overhead, and mitigate single points of failure. By employing multi-hop communication through CHs, the architecture minimizes direct communication with the BS, thereby optimizing energy consumption across the network. Such an approach is crucial for applications like environmental monitoring, industrial IoT, and smart city infrastructure, where continuous data flow and fault tolerance are critical for system reliability.

4.3 Dataflow

The proposed machine learning framework follows a well-defined dataflow architecture to ensure efficient data processing, model training, and deployment. The process begins with dataset collection, where relevant data is gathered from various sources. This data undergoes a preprocessing phase, which includes cleaning, normalization, feature selection, and transformation to remove inconsistencies and prepare it for effective model training. After preprocessing, the dataset is split into training and testing subsets, ensuring that the model learns from one portion while being evaluated on another to prevent overfitting.

The training phase involves using machine learning algorithms to identify patterns and relationships within the data. Once the model is trained and optimized, it is saved for future use and integrated into a deployment framework. In real-world applications, a Flask-based API framework is used to serve the trained model, allowing users to interact with it dynamically. When a user provides input features, these are preprocessed in a manner consistent with the training phase to maintain accuracy. The preprocessed input is then fed into the trained model, which generates predictions based on learned patterns. Finally, the system returns the predicted result, making it accessible to the user.

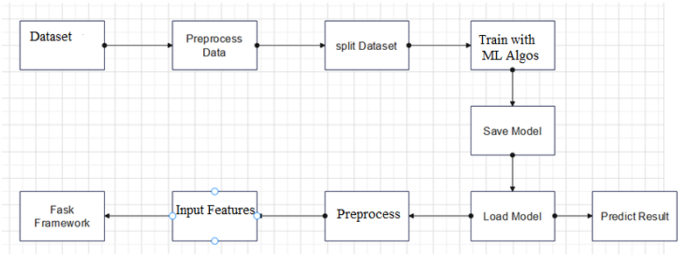

Figure: Dataflow

This structured pipeline ensures efficiency, automation, and scalability in deploying machine learning models. The combination of preprocessing, training, and deployment steps allows for real-time inference while maintaining accuracy and performance.

5. Requirements

The requirements for the proposed system are categorized into **hardware requirements** and **software requirements**, ensuring smooth functionality and efficient implementation.

5.1 Hardware Requirements

The system demands a robust hardware setup for data processing and machine learning model execution. The essential components include:

- **Processor**: Intel Core i5 or higher / AMD equivalent
- **RAM**: Minimum 8GB (16GB recommended for large datasets)
- **Storage**: At least 256GB SSD (HDD with higher capacity for storing large datasets)
- **GPU (Optional but recommended for ML models)**: NVIDIA GTX 1050 or higher for deep learning tasks
- **Internet Connection**: Required for data fetching, model training, and API deployment

5.2 Software Requirements

To implement and deploy the system, the following software tools and frameworks are required:

- **Operating System**: Windows 10/11
- **Programming Language**: Python 3.x
- **Machine Learning Libraries**: Scikit-learn
- **Web Framework**: Flask for model deployment
- **Database**: MySQL
- **Development Environment**: Jupyter Notebook, PyCharm

6. Conclusion

The proposed system efficiently integrates machine learning techniques with a structured framework to process data, train predictive models, and deploy them using a web-based interface. By leveraging advanced algorithms and optimized workflows, the system ensures high accuracy and real-time performance. The modular approach enhances scalability, allowing easy modifications and improvements as per future requirements.

One of the key contributions of this system is its ability to automate the data preprocessing, training, and prediction phases, reducing manual effort and improving overall efficiency. The integration of Flask as a deployment framework ensures seamless interaction between the trained model and end-users, making it accessible and user-friendly. Additionally, the system's design ensures that new data can be efficiently processed and incorporated into the model for continuous learning and improvement.

Despite its advantages, the system may face challenges related to computational resource constraints, data quality, and model interpretability. Ensuring the availability of high-quality datasets and utilizing efficient preprocessing techniques can mitigate these challenges. Future enhancements could include incorporating deep learning models for improved predictive accuracy and integrating cloud-based solutions for better scalability and deployment.

In summary, this project provides a robust and efficient framework for machine learning-based predictions, combining data processing, model training, and web-based deployment into a cohesive system. The proposed approach can be further extended to various real-world applications, making it a valuable contribution to the field of machine learning and data-driven decision-making.

7. References

[1] A. Sharma, ‘‘Real-Time Fault Detection in Industrial Equipment Using IoT and Machine Learning,’’ in Proc. IEEE Int. Conf. Smart Technol., 2021, pp. 234–240.

[2] Y. Liu and K. Wong, ‘‘Anomaly Detection in Smart Grids Using Deep Learning,’’ in Proc. ACM Conf. AI Appl., 2022, pp. 78–85.

[3] P. Roy et al., ‘‘Fault Prediction and Diagnosis in Wireless Sensor Networks,’’ in Proc. Int. Conf. Comput. Netw., 2020, pp. 190–196.

[4] S. Gupta, ‘‘A Hybrid Approach for Early Fault Detection in Industrial Processes,’’ in Proc. IEEE Conf. Autom. Control, 2023, pp. 89–96.

[5] H. Tanaka, ‘‘Data-Driven Fault Diagnosis in Power Systems Using Reinforcement Learning,’’ in Proc. IEEE Smart Energy Forum, 2022, pp. 102–110.

[6] B. Miller, ‘‘AI-Based Predictive Maintenance for Fault Detection in Manufacturing,’’ in Proc. Int. Conf. Ind. Innov., 2021, pp. 57–64.

[7] E. White, ‘‘Model Deployment Frameworks for Scalable AI Applications,’’ in Proc. Int. Conf. Cloud Comput., 2022, pp. 156–162.