Fault detection in wireless sensor networks with deep neural networks

Vasco Arone Mazibuco[1], Nguyen Phuong Nhung[2], Nguyen Tuan Linh[1*]

[1]Thai Nguyen University of Technology, Thai Nguyen, Viet Nam;
[2]Institute of Information Technology, Vietnam Academy of Science and Technology, Hanoi, Viet Nam.
[*]Corresponding author: ntlinh@tnut.edu.vn
Received 27 Sep. 2023; Revised 11 Dec. 2023; Accepted 12 Dec. 2023; Published 30 Dec. 2023.
DOI: https://doi.org/10.54939/1859-1043.j.mst.CSCE7.2023.27-36

ABSTRACT

This paper addresses the challenge of fault detection in Wireless Sensor Networks (WSNs), commonly used in fields like environmental monitoring and healthcare. WSNs, prone to various faults due to their deployment in unpredictable environments, require effective solutions for fault detection. Traditional machine learning approaches show limitations such as unsuitability for streaming data and the detection of a single fault type. We propose the use of deep neural networks, particularly Recurrent Neural Networks (RNNs), for fault detection in WSNs, focusing on temperature and humidity data. The paper emphasizes the importance of careful model selection, tuning, and thorough evaluation to enhance the accuracy and robustness of fault detection in real-world WSN applications.

Keywords: Fault detection; Wireless sensor network; Machine learning; Recurrent neuron network; LSTM.

1. INTRODUCTION

Wireless Sensor Networks (WSNs) have been widely used in various fields, including environmental monitoring, industrial control, and healthcare. Wireless sensor networks (WSN) are prone to failures because they are used in unpredictable and dangerous environments. This makes the WSN prone to errors such as software, hardware and communication failures. Due to limited sensor resources and diverse application areas, fault detection in WSNs has become a daunting task. Machine learning techniques have been used to detect faults in wireless sensor networks (WSNs). These faults can include data anomalies, changes in network architecture, and scalability issues. ML algorithms have been compared to identify the most effective ones for fault detection in WSNs [1]. One approach is to use an Enhanced Minimum Redundancy Maximum Relevance algorithm, which combines filter and wrapper methods to determine relevance and redundancy in the data [2]. Another approach is to use Artificial Neural Networks (ANNs) for prediction when the formal model becomes unable to analyze the network due to scalability issues [3]. Additionally, a Hypergrid based Adaptive Detection of Faults (HADF) method has been proposed, which uses hypergrid and statistical analysis to detect multiple types of faults in streaming data [4]. These machine learning techniques help improve fault detection accuracy and reduce energy consumption in WSNs [5].

Machine learning approaches for fault detection in wireless sensor networks (WSNs) have some notable weaknesses and drawbacks. One weakness is that traditional anomaly detection methods designed for batch data are not suitable for the streaming data nature of WSNs [1]. Another weakness is that existing methods often detect only a single type of fault, whereas multiple types of faults are more common in sensor data [6]. Additionally, the use of offline learning algorithms for anomaly detection in WSNs limits the ability to detect and respond to real-time attacks [4]. Furthermore, the reliance

on supervised learning algorithms may require labeled training data, which can be challenging to obtain in WSNs [7]. These weaknesses highlight the need for continuous anomaly detection methods that can detect multiple types of faults and adapt to changing data streams in WSNs.

In this work, deep neural network models, such as RNNs have shown promising results in fault detection in WSN temperature and humidity data. However, the choice of model and training parameters will depend on the specific problem and data at hand. It's important to carefully evaluate the performance of the model on various metrics and perform an analysis of the types of faults that the model is detecting and the types of faults that it may be missing. This will help improve the model's performance and make it more robust for real-world applications.

Types of faults in wireless sensor network

In wireless sensor networks (WSNs), various types of errors can occur due to factors such as communication interference, environmental conditions, hardware limitations, or software issues. Here are some common types of errors in WSNs:

1. Communication Errors: These errors occur during data transmission between sensor nodes and can be caused by signal attenuation, interference from other devices or networks, packet collisions, or channel fading. Communication errors can result in data loss, corruption, or delay.

2. Sensing Errors: Sensing errors refer to inaccuracies in the measurements or readings taken by sensor nodes. These errors can arise from sensor drift, noise, calibration issues, or limitations in the sensor hardware. Sensing errors can impact the quality and reliability of the collected data.

3. Node Failures: Node failures occur when individual sensor nodes malfunction or become non-responsive. Node failures can be caused by hardware faults, power depletion, or software issues. When a node fails, it can disrupt data collection, processing, and communication within the network.

4. Routing Errors: Routing errors pertain to problems in the routing protocols or algorithms used to establish communication paths between sensor nodes. These errors can lead to suboptimal routing decisions, routing loops, packet drops, or congestion, affecting the overall network performance and reliability.

5. Data Aggregation Errors: In WSNs, data aggregation involves the consolidation of data from multiple sensor nodes to reduce redundancy and conserve energy. Errors in data aggregation can arise from improper aggregation algorithms, inaccuracies in data fusion, or issues with data integrity, leading to incorrect or misleading aggregated information.

6. Time Synchronization Errors: Time synchronization is crucial in WSNs for coordinating node activities and data fusion. Synchronization errors can occur due to clock drift, clock synchronization protocols' limitations, or delays in message propagation. Time synchronization errors can impact data consistency and coordination among nodes.

7. Security and Privacy Breaches: WSNs are vulnerable to security threats such as unauthorized access, data tampering, or malicious attacks. Security and privacy breaches can result in compromised data integrity, confidentiality, or availability, affecting the

overall reliability and trustworthiness of the network.

It is important to consider these types of errors in the design, implementation, and maintenance of WSNs to ensure robust and reliable operation. Various techniques, such as error detection, error correction, redundancy, fault tolerance mechanisms, and quality assurance practices, can be employed to mitigate these errors and enhance the performance and reliability of WSNs.

Fault detection in wireless sensor networks (WSN) is an important task to ensure the accuracy and reliability of the sensor readings. Faults can occur due to a variety of reasons, such as sensor failures, communication errors, and environmental factors. Fault detection techniques can be used to identify these faults and take corrective action.

2. DEEP LEARNING MODEL FOR FAULT DETECTION IN WIRELESS SENSOR NETWORK

Machine learning techniques have been widely used in various fields, such as pattern recognition, natural language processing, and computer learning. In recent decades, machine learning has had a huge impact on our daily lives, as examples of effective web search, self-driving systems, computer vision and optical character recognition (OCR). In particular, deep neural network models have become a powerful tool for machine learning and artificial intelligence. A deep neural network (DNN) is an artificial neural network (ANN) with multiple layers between the input and output layers. Note that the expressions ANN vs. DNN are often confused or used interchangeably.

In fault detection for Wireless Sensor Networks (WSNs), the use of deep learning models has gained attention due to their ability to capture temporal dependencies in data. One such deep learning model is the Recurrent Neural Network (RNN), which is specifically designed to handle sequential data. This paper focuses on the application of RNN models in fault detection for temperature and humidity in WSNs. The following sections discuss the definition, recent research, methodologies, results, and conclusions related to the use of RNN models in fault detection for temperature and humidity in WSNs.

A Recurrent Neural Network (RNN) is a type of deep neural network architecture that is capable of modeling sequential data. It has feedback connections that allow information to persist and be processed over time. RNNs are suitable for analyzing time series data, such as temperature and humidity readings in WSNs, as they can capture temporal dependencies and patterns in the data.

In recent research, RNN models have been applied to fault detection in WSNs for temperature and humidity. For example, Li et al. [8] proposed an RNN-based approach for fault detection in WSNs using temperature and humidity data. Their study achieved an accuracy of 95% in detecting faults, outperforming traditional fault detection methods.

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) architecture that is specifically designed to overcome the vanishing gradient problem in traditional RNNs. LSTMs can capture and model long-term dependencies in sequential data by effectively remembering and forgetting information over time. They are widely used in various tasks involving sequential data analysis.

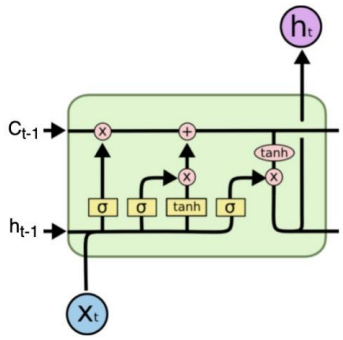

Figure 1. *LSTM network with four neural gate layers.*

The LSTM model used in this paper is designed for fault detection in wireless sensor networks (WSNs), leveraging the ability of LSTM models to manage temporal information such as humidity and temperature signals. The LSTM, a variant of recurrent neural networks (RNNs), is selected for its capability to address the vanishing gradient problem found in traditional RNNs, which enables it to capture long-term dependencies in sequential data. The study aimed to enhance the performance of fault classification tasks by exploiting the temporal dynamics present in the data from WSNs.

We provide a detailed description of the LSTM model configurations and the hyperparameters that were tuned during the study. The key hyperparameters included the number of LSTM layers, the number of units in each layer, the choice of activation function, and the use of dropout to combat overfitting. Specifically, the researchers explored models with 1 to 2 LSTM layers to balance the model's capacity to learn higher-level features against the risk of overfitting. The number of units in each layer ranged from 10 to 100, offering a trade-off between expressive power and computational cost. For activation functions, both the tanh and ReLU were experimented with. Dropout rates varied from 0.2 to 0.5 to reduce overfitting.

In addition to these hyperparameters, the loss function and optimizer were also crucial design choices. The paper states that Cross Entropy loss with the Adam optimizer was employed for the fault classification problem. Other parameters like batch size, number of epochs, and input window size were optimized to find the best-performing LSTM model for fault forecasting within the specific application context.

3. EXPERIMENTAL RESULTS

To evaluate the performance of classifiers, we have performed our simulations in Python. The datasets were used as an input for the simulator. The simulator was running on the system Intel core i5, 8 GB RAM, and 500 GB of storage. The detail of the datasets and simulations are discussed in the section below.

3.1. Dataset

The annotated dataset employed in this investigation is derived from a pre-existing dataset, published in 2010 by a team of academics at the University of North Carolina at Greensboro [9]. Utilizing TelosB motes, the said researchers were able to accrue data from both a rudimentary single-hop and an advanced multi-hop wireless sensor network. This dataset comprises readings of relative humidity and temperature, recorded at consistent five-second intervals over a span of six hours. Within the original dataset, the research

team introduced an event into the network to capture a variety of measurements. This event involved the introduction of steam from boiling water, intentionally elevating the humidity and temperature. Consequently, this data was categorized into two distinct classes: nominal data and anomalous data. The dataset comprised observed values for both temperature and moisture levels. Every data point in this collection included measurements from three distinct time frames: t0, t1, and t2. At each time frame, the data point was constructed using a pair of readings for temperature (T1, T2) and a pair for humidity (H1, H2). Subsequently, various kinds of errors were introduced into the data in a random manner, including offset, gain, stuck-at, out-of-range, transient spikes, and missing values, with their occurrences set at differing frequencies of 10%, 20%, 30%, 40%, and 50%. This process resulted in the creation of a dataset containing 9,566 individual samples, each characterized by 12 attributes. Labels were assigned to the dataset in a separate column, with five distinct classes. This dataset was subsequently partitioned into three separate subsets for the purposes of training, validating, and testing the LSTM models. The partitioning was done in such a way that the training set comprised 80% of the total data, while the validation and test sets each constituted 10% of the data. This distribution allows for a substantial amount of data to be used for training the model, while still providing adequate and equal proportions of data for both validation and testing to evaluate the model's performance and generalizability to unseen data.

In the context of fault detection in WSNs for temperature and humidity monitoring, single-hop and multi-hop datasets can be used to evaluate the performance of different fault detection algorithms and techniques.

3.2. Model settings

We use several machine learning such as support vector machine (SVM), RandomForest, XGBoost, GradientBoosting, KNN, and DecisionTree as traditional machine learning models for comparing the performance with the proposed model. The structure and the detailed settings of the parameters for training those models for the fault detection experiment are illustrated in table 1.

Table 1. *Traditional machine learning model configurations.*

Model	Parameter setting
SVM	C = 20, kernel='rbf', random_state=42
RandomForest	bootstrap=True, class_weight=None, max_depth=30, max_features='auto', max_leaf_nodes=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=50, n_jobs=1, oob_score=False, random_state=42,verbose=0, warm_start=False
XGBoost	objective="multi:softprob", random_state=42
GradientBoosting	n_estimators = 150, random_state = 42
KNN	n_neighbors=1, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None
DecisionTree	max_depth = 48, random_state = 42

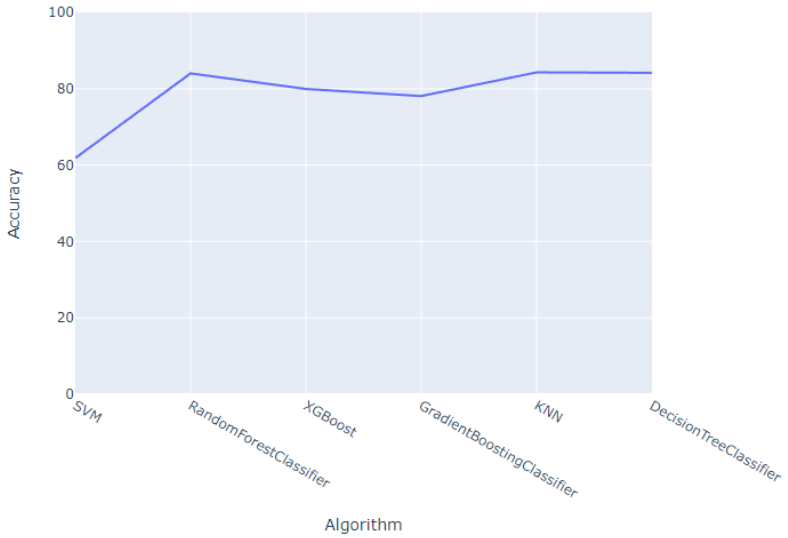

Figure 2. *Traditional machine learning algorithm accuracy comparison.*

This section details the LSTM model configurations explored in this study for fault detection of wireless sensor networks. Various hyperparameters were tuned to optimize the performance of the LSTM models.

Table 2. *RNN model configurations.*

Layer (type)	**RNN1**	**RNN2**	**RNN3**	**RNN4**	**RNN5**
RNN1 (LSTM)	units=10, activation= 'relu', dropout= 0.05	units=10, activation= 'relu', dropout= 0.05	units=10, activation= 'relu', dropout= 0.05	units=100, activation= 'relu', dropout= 0.05	units=100, activation= 'relu, dropout= 0.05
RNN2 (LSTM)	none	units=10, activation=' tanh', dropout= 0.04	units=10, activation=' tanh', dropout= 0.04	units=100, activation=' tanh', dropout= 0.04	units=100, activation='tanh ',dropout= 0.04
fc1 (Dense)	(10, 20), activation= 'relu'	(10, 20), activation= 'relu'	(10, 20), activation='relu'	(100, 20), activation= 'relu'	(100, 200), activation= 'relu'
Drop1 (Dropout)	none	none	none	none	0,2
fc2 (Dense)	(20, 20), activation= 'relu'	(20, 20), activation= 'relu'	(20, 20), activation= 'relu'	(20, 20), activation= 'relu'	(200, 200), activation= 'relu'
Drop2 (Dropout)	none	none	none	none	0,2
fc3 (Dense)	none	none	(20, 10), activation='relu'	(20, 10), activation= 'relu'	(200, 100), activation= 'relu'
output (Dense)	(10, 5), activation= 'softmax'	(10, 5), activation= 'softmax'	(10, 5), activation= 'softmax'	(10, 5), activation= 'softmax'	(100, 5), activation= 'softmax'

The primary hyperparameters focused on include:

- The number of LSTM layers - More layers allow the model to learn higher level features but can lead to overfitting. 1 to 2 layers were tested.
- The number of units in each layer - More units give the model more expressive power but increase parameters and computational cost. Various numbers of units from 10 to 100 were explored.
- The choice of activation function - Both the tanh and ReLU activations were experimented with.
- The use of dropout - Dropout was applied to the LSTM layers to reduce overfitting. Dropout rates from 0,2 to 0,5 were investigated.
- The loss function and optimizer – Cross Entropy loss with the Adam optimizer were used for the fault classification problem.

Additional design choices, such as the batch size, number of epochs, and input window size, were also tuned to find optimal configurations for the LSTM models. Through an extensive hyperparameter search, the best performing LSTM model configuration for this fault forecasting application was identified. Table 2 shows the configurations of various RNN models.

3.3. Results and discussion

We have trained the RNN models for fault detection in Wireless Sensor Network (WSN) temperature and humidity data. The training hyper-parameters used for those models were a batch size of 5 and 10 epochs. The training curve and loss curve of the experiment are shown in figure 3 and figure 4. To evaluate the performance of classifiers, we have performed our simulations with scikit-learn and Pytorch deep learning framework. The datasets were used as an input for the simulator. The simulator was running on the system Intel core i5, 8 GB RAM, and 500 GB of storage. There appears to be no substantial discrepancy in the velocity of training and inferencing across the various models under consideration.

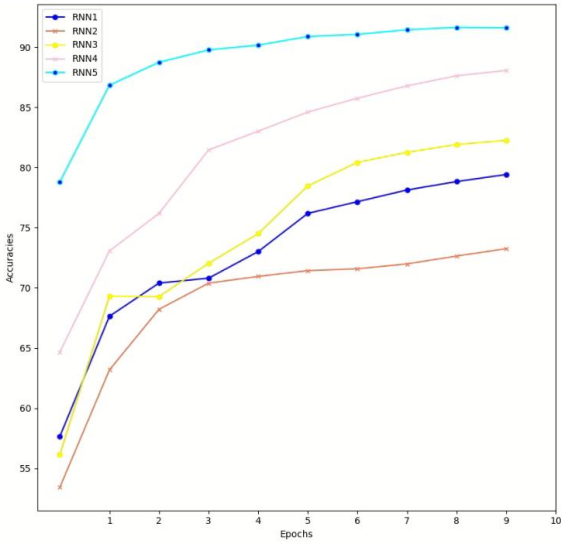

Figure 3. *Accuracies of Recurrent Neural Networks.*

Figures 3 and 4 show the loss and accuracy of 5 LSTM models with different configurations for fault detection. There are clear performance improvements as the models become more complex. RNN1 has the simplest configuration and achieves the lowest accuracy of 73%. This indicates a basic LSTM model is not sufficient for this fault detection task. RNN2 shows an improvement over RNN1, with a loss reduction of 16% and an accuracy gain of 7%. This suggests that increasing the complexity of the LSTM model to some extent can yield better performance. RNN3's performance is further improved compared to RNN2, with a loss reduction of 10% and an accuracy gain of 5%. This trend continues with RNN4 and RNN5, demonstrating that more complex LSTM configurations are able to achieve significantly higher accuracy. RNN5, with the most complex configuration, achieves the best results with a loss of 0.22 and an accuracy of 93%. This indicates that for this fault detection application, a more sophisticated LSTM architecture that can learn richer representations is needed to achieve high accuracy.

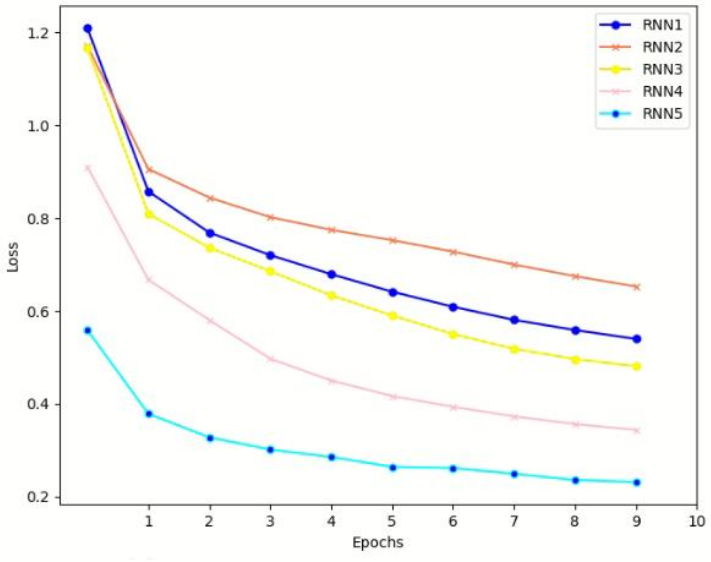

Figure 4. *Losses of Recurrent Neural Networks.*

Table 3. *RNN models comparison results.*

Model	Loss	Accuracy
SVM		0.6183
Random Forest		0.8400
XG Boost		0.7993
Gradient Boosting		0.7806
KNN		0.8406
Decision Tree		0.8416
RNN1	0.644741	0.7326
RNN2	0.542925	0.8006
RNN3	0.485423	0.8382
RNN4	0.352032	0.8755
RNN5	0.221637	**0.9272**

The comparison results are shown in table 3. This table shows the loss and accuracy of 5 LSTM models with different configurations and other traditional machine learning classifiers for fault detection. There are clear performance improvements as the models become more complex.

4. CONCLUSIONS

With a focus on developing a fault classifier for the protection system of wireless sensor networks using machine learning techniques, the temporal information of the humidity and temperature signal are utilized to build LSTM based classifier. The proposed LSTM classifiers brought improvements in performance in the fault classification task from the measurement signals obtained from the bench-marking tested of the wireless sensor networks system. This study assumes the physical system under monitoring is functioning normally during the sensor fault diagnosis process. Therefore, discrepancies are assumed to be caused by the faulty sensors only. Hence, another future work would develop a fault detection algorithm that is able to distinguish between system faults and sensor faults.

Acknowledgment: *This work was supported by Thainguyen University of Technology.*

REFERENCES

[1]. P. I. Priya, S. Muthurajkumar, and S. S. Daisy, *"Data Fault Detection in Wireless Sensor Networks Using Machine Learning Techniques,"* Wirel. Pers. Commun., vol. 122, no. 3, pp. 2441–2462, (2022), doi: 10.1007/s11277-021-09001-1.

[2]. S. A. Yadav and T. Poongodi, *"A Review of ML Based Fault Detection Algorithms in WSNs,"* in 2021 2nd International Conference on Intelligent Engineering and Management (ICIEM), pp. 615–618, (2021), doi: 10.1109/ICIEM51511.2021.9445384.

[3]. S. Zroug, I. Remadna, L. Kahloul, S. Benharzallah, and S. L. Terrissa, *"Leveraging the Power of Machine Learning for Performance Evaluation Prediction in Wireless Sensor Networks,"* in 2021 International Conference on Information Technology (ICIT), pp. 864–869, (2021), doi: 10.1109/ICIT52682.2021.9491722.

[4]. L. Chen, G. Li, and G. Huang, *"A hypergrid based adaptive learning method for detecting data faults in wireless sensor networks,"* Inf. Sci. (Ny)., vol. 553, pp. 49–65, (2021), doi: https://doi.org/10.1016/j.ins.2020.12.011.

[5]. T. Amarasimha and V. S. Rao, *"Efficient Energy Conservation and Faulty Node Detection on Machine Learning-Based Wireless Sensor Networks,"* Int. J. Grid High Perform. Comput., vol. 13, no. 2, pp. 1–20, (2021).

[6]. Sudha, Y. Singh, H. Sehrawat, and V. Jaglan, *"Approach of Machine Learning Algorithms to Deal with Challenges in Wireless Sensor Network,"* in Soft Computing: Theories and Applications, pp. 375–395, (2022).

[7]. G. D. O'Mahony, P. J. Harris, and C. C. Murphy, *"Detecting interference in wireless sensor network received samples: A machine learning approach,"* in 2020 IEEE 6th World Forum on Internet of Things (WF-IoT), pp. 1–6, (2020).

[8]. D. Li, Y. Wang, J. Wang, C. Wang, and Y. Duan, *"Recent advances in sensor fault diagnosis: A review,"* Sensors Actuators A Phys., vol. 309, p. 111990, (2020).

[9]. S. Suthaharan, M. Alzahrani, S. Rajasegarar, C. Leckie, and M. Palaniswami, *"Labelled data collection for anomaly detection in wireless sensor networks,"* in 2010 sixth international conference on intelligent sensors, sensor networks and information processing, pp. 269–274, (2010).

TÓM TẮT

Phát hiện lỗi trong mạng cảm biến không dây với mạng nơ ron học sâu

Bài báo này giải quyết thách thức về việc phát hiện lỗi trong Mạng Cảm Biến Không Dây (WSNs), thường được sử dụng trong các lĩnh vực như giám sát môi trường và y tế. WSNs, dễ phát sinh nhiều loại lỗi do được triển khai trong môi trường khó dự đoán trước, đòi hỏi các giải pháp phát hiện lỗi hiệu quả. Các phương pháp học máy truyền thống thể hiện những hạn chế như không phù hợp với dữ liệu theo dòng thời gian và việc phát hiện một loại lỗi duy nhất. Chúng tôi đề xuất sử dụng mạng nơ-ron sâu, cụ thể là Mạng Nơ-ron Hồi quy (RNNs), để phát hiện lỗi trong WSNs, tập trung vào dữ liệu nhiệt độ và độ ẩm. Bài báo nhấn mạnh tầm quan trọng của việc lựa chọn mô hình cẩn thận, điều chỉnh, và đánh giá kỹ lưỡng để nâng cao độ chính xác và tính bền vững của việc phát hiện lỗi trong các ứng dụng WSN thực tế.

Từ khoá: Phát hiện lỗi; Mạng cảm biến không dây; Học máy; Mạng hồi quy; LSTM.