An LSTM-Based Outlier Detection Approach for IoT Sensor Data in Hierarchical Edge Computing

Somia Bibi, Chafiq Titouna, Faiza Titouna, Farid Naït-Abdesselam

An LSTM-based outlier detection approach for IoT sensor data in hierarchical Edge Computing

Somia Bibi * Chafiq Titouna † Faiza Titouna * ‡Farid Naït-Abdesselam
*University of Batna 2, Algeria
†LIGM, ESIEE Paris, University Gustave Eiffel, France
‡ University of Missouri Kansas City, USA

***Abstract*—Outlier detection in sensor data has recently gained significant recognition, particularly with the proliferation of wireless sensor networks (WSNs) and the Internet of Things (IoT). Several challenges face outlier detection in WSNs and IoTs, including sensor nodes' limited energy and processing capabilities and high communication costs. This paper presents a novel deep learning-based outlier detection approach for IoT sensor data in hierarchical edge computing. First, we proposed a hierarchical edge computing framework to save energy, provide load balance, and low latency data processing at sensor ends. Then, we designed an outlier detection algorithm that resides on each edge server. The proposed algorithm consists of two modules: a predictor model and an outlier detector. The predictor module uses Long Short-Term Memory (LSTM) networks to predict the subsequent data measurements of sensor nodes. The predicted values are then passed to an outlier detector module, which decides whether a data point is an outlier.**

***Index Terms*—Internet of Things, Wireless Sensor Networks, Edge Computing, outlier detection, Deep learning, Short-Term Memory (LSTM).**

I. Introduction

The Internet of Things (IoT) [1] is an emerging technology that has aroused widespread interest from both research and industry communities in the past few years. An IoT can broadly be described as a network of interrelated physical objects (e.g., cars, houses, laptops, thermostats, watches) that can transfer and exchange data through a wireless network without explicit human intervention. These devices are typically equipped with sensors, actuators, processors, and other embedded technologies that allow them to sense, collect, and share data about the environment in which they are operated. IoTs are being used in diverse fields, including environment monitoring [2], medicine and health care [3], transportation [4], agriculture [5], intelligent buildings [6], logistics [7], and manufacturing [8].

Wireless Sensor Networks (WSNs), which consist of many wirelessly interconnected sensor nodes and one or more base stations (sink nodes), are among the primary building blocks that enable the operation of IoT applications. A sensor node in a WSN is typically composed of a microcontroller, radio transceiver, memory, power supply, and one or several sensors [9].

Wireless sensor nodes are usually constrained in terms of energy, memory, and processing. Hence they are unable to carry out complex computational tasks. Computation offloading [10] is a viable technique that can prolong the lifetime of sensor nodes by relocating computational tasks from these devices to resource-abundant servers such as clouds.Yet, cloud servers are usually insufficient to meet the high-performance requirements of some IoT applications due to several reasons such as high latency, limited bandwidth, and security issues. Edge computing, where computing resources are distributed closer to end nodes, is a new paradigm that aims to overcome some of the issues faced by cloud computing. It can significantly save bandwidth, reduce communication latency, improve security, and provide real-time services.

Due to the effects of hostile environments and the innate limitations in sensor capabilities, the data collected from IoT and WSNs is often uncertain and prone to errors and faults. Outliers are one of the most common faults that may severely affect the quality of sensory data. Outliers, also termed anomalies, are defined in various ways depending on the discipline in which they are being studied. Barnett and Lewis [11] defined an outlier as "An observation (or subset of observations) which appears to be inconsistent with the remainder of that set of data". Similarly, the authors in [12] have defined an outlier as "a data point which is significantly different from other data points, or does not conform to the expected normal behaviour, or conforms well to a defined abnormal behaviour". In the context of WSN and IoT, outliers are those sensor readings that do not follow the normal pattern of sensed data. Outliers may arise from three primary sources: (i) Errors caused by impaired readings generated by faulty sensors, (ii) Events that occur due to a sudden or unexpected change in the sensed environment (e.g., forest fire, earthquakes,air pollution) , (iii) Malicious attacks such as denial of service, black hole, and sinkhole attacks.

Regardless of the application field or the outlier source, detecting outliers in sensor data is essential for ensuring high data quality and reliability, enhancing the system's security, and maximizing the network's lifetime.

In the last decade, several approaches have been proposed to address the outlier detection [13]–[15] (anomaly detection) problem in sensor data. However, most of these approaches are computationally costly or incur high communication overhead.

The main contributions of this paper are as follows:

1) A hierarchical edge computing model is proposed, which relocates computing workloads from cloud centers to edge servers to provide more processing capabilities and

low latency analysis on the premise of enhancing the system's performance and lowering the energy consumption.
2) An LSTM-based outlier detection approach for sensor data is proposed. This approach is deployed and executed at each edge node. The proposed approach consists of two modules: a predictor model and an outlier detector. The predictor model employs LSTM to forecast the subsequent sensor data measurements. The outlier detector identifies outliers by computing the distances between the predicted and the actual values.

The remainder of this paper is organized as follows. Section II reviews the literature related to outlier detection in sensor data. Section III defines in detail the proposed system architecture. Section IV provides a detailed description of our proposed scheme. The performance evaluation and the results are presented in Section V. Finally. We conclude the paper in Section VI.

II. Related works

As mentioned above, several methods have been proposed to deal with the outlier detection problem in WSNs. These methods can be grouped into five main categories: (i) statistical-based techniques, (ii) classification-based techniques, (iii) nearest neighbor-based techniques, (iv) clustering-based techniques and (v) artificial intelligence-based techniques [16].

Recently, deep learning algorithms have been leveraged for outlier detection in wireless sensor networks. The authors in [17] employed autoencoder neural networks to detect anomalies in WSNs. They designed a two-part algorithm that resides on sensors where outliers are identified locally without communicating with a central IoT cloud or nearby sensors and on the cloud to detect global outliers. The performance of the proposed method was evaluated on a real sensor dataset, and the obtained results were very promising.

In [18], an artificial neural network-based outlier detection model is proposed for WSNs in smart buildings. In this work, an ANN model is designed to forecast the temperature values of each sensor node. A temperature data is regarded as an outlier if the difference between the predicted and the actual value exceeds a certain threshold. The authors of have also designed an optimized ANN model to increase detection accuracy in the case of a significant environmental temperature change. The experimental results showed that the two ANN-based models could effectively forecast the temperature values in WSNs, and the optimized ANN performed better when the environment temperature changed significantly.

In [19], the authors proposed a new scheme to detect outliers in sensor networks by using a bidirectional Long Short-Term Memory Recurrent Neural Network (BLSTM-RNN) and a modified version of the auto-encoder called Smooth Auto-Encoder (SmAE). SmAE is used to learn strong plus discriminative feature representations. BLSTM-RNNs are employed for maturity voting for collective outlier detection. The experimental results on the Intel Berkeley Research Laboratory (IBRL) dataset show that the proposed scheme reaches higher accuracy and recall than existing techniques.

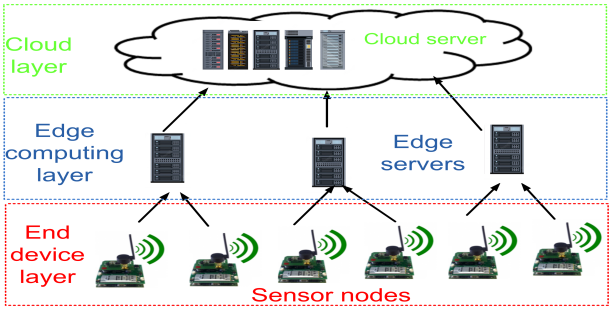


Fig. 1. System architecture

In 2022, Zhang et al. [20] designed a scheme that considers temporal and spatial dimensions to detect outliers in multimodal WSN data flows based on Graph neural networks (GNN) and attention mechanism. First, the temporal and modal correlation features derived from each sensor are combined into a single vector representation using Graph Attention Network. Afterward, it is aggregated with the spatial features representing the nodes' spatial position relationship. Finally, the current time-series data of sensor nodes are forecasted, and anomalous states are detected according to the fusion features. The simulation results have shown that the proposed scheme has a better detection performance than the traditional graph convolution network method combined with LSTM.

III. System architecture

The architecture of our proposed system is illustrated in Figure 1, it can be seen as a three-tiered structure consisting of a central cloud server in the first tier, multiple edge servers in the second tier, and multiple sensor nodes in the third tier. Their roles are described as follows:

- **Sensor nodes:** Sensor nodes are small and low-power devices that are equipped with a microcontroller, a radio transceiver, a memory, a battery, and one or more sensors. They are responsible for periodically collecting data from their surroundings and transmitting it to nearby edge servers via a wireless medium.
- **Edge servers:** Edge servers are powerful computers that are geographically distributed around the sensor nodes. They are intermediaries between the end devices (sensor nodes) and the cloud server. Each edge server is responsible for two main tasks: (i) predicting the subsequent measurements of sensor nodes, (ii) perform outlier detection and send the results to the cloud server.
- **Cloud server:** The cloud server stores and analyzes the data received from the edge servers and makes decisions accordingly.

IV. Proposed scheme

In this section, we will describe in detail our proposed scheme to detect outliers in sensor data. The proposed method consists of two modules: an LSTM-based prediction model that forecasts the subsequent sensor measurements based on the previous records and an outlier detector that decides whether a data measurement is an outlier.

A. Predictor model

The predictor module of our proposed approach is based on LSTMs. LSTMs [21] are a specialized type of recurrent neural network (RNN) which have been widely used in different domains, such as natural language processing (NLP) and big data prediction. They were designed to overcome the vanishing and exploding gradients problem experienced by traditional RNNs. A conventional LSTM consists of an input, hidden layer (LSTM), and output layer. The LSTM layer comprises several memory blocks, each containing one or more memory cells and three multiplicative gates: forget, input, and output (As depicted in Figure 2):

- The forget gate is responsible for deciding whether the information from the previous timestamp is to be kept or thrown away.
- The input gate decides whether the memory cell is updated by combining the second sigmoid layer's output and the tanh layer's output.
- The cell state is responsible for long-term memory and is calculated based on the previous cell state and the outputs of the forget and input gates.
- The output gate specifies which data from the current cell state will be used as output.

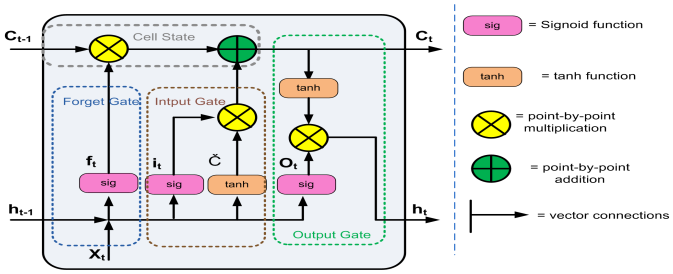


Fig. 2. Structure of the LSTM cell.

The following equations can define the entire computation of long short-term memory networks:

$$f^{(t)} = \sigma(W_f[h^{(t-1)}, x^{(t)}] + b_f) \tag{1}$$

$$i^{(t)} = \sigma(W_i[h^{(t-1)}, x^{(t)}] + b_i) \tag{2}$$

$$\widetilde{C}^{(t)} = tanh(W_C[h^{(t-1)}, x^{(t)}] + b_C) \tag{3}$$

$$C^{(t)} = f^{(t)} \odot C^{(t-1)} + i^{(t)} \odot \widetilde{C}^{(t)} \tag{4}$$

$$o^{(t)} = \sigma(W_o[h^{(t-1)}, x^{(t)}] + b_o) \tag{5}$$

$$h^{(t)} = tanh(C^{(t)}) * o^{(t)} \tag{6}$$

Table I lists the main symbols and notations used in the above equations. In the present study, we have designed three LSTM-based predictor models: the temperature forecasting model, the humidity forecasting model, and the voltage forecasting model.

TABLE I
SUMMARY OF SYMBOLS AND NOTATIONS

Symbols and Notations	Description
$f^{(t)}, i^{(t)}, o^{(t)}$	The forget, input, and output gate at time t, respectively.
σ	Logistic sigmoid function
W_f, W_i, W_o	Weight matrices of the forget, input and output gates, respectively.
b_f, b_i, b_o, b_C	The biases of the forget, input, output gates, and the cell state, respectively.
$h^{(t-1)}, h^{(t)}$	The hidden state of the previous and timestamp, respectively.
$x^{(t)}$	The current input
$C^{(t)}$	The current cell state
$\widetilde{C}^{(t)}$	The new memory cell state candidate
$\odot$	Element-wise (Hadamard) product.

B. Outlier detector

Once the LSTM predictor model forecasts the next sensor data measurement, this module detects outliers at each time step. The predicted data measurement is passed to this module, and then an outlier score is assigned to a given sensor reading based on the distance between this reading and the value predicted by the LSTM model. Outlier scores are computed using the Euclidean distance given in Equation (7). Data measurements with scores greater than a given threshold are considered outliers. The threshold value is calculated using the equation

$$O(y_t) = d_{(y_t, y'_t)} = \sqrt{(y_t - y'_t)^2} \tag{7}$$

Where:

$O(y_t)$: denotes the outlier score of a data measurement.
y_t : is the actual value.
y'_t : is the predicted value.

$$\lambda = O(minOutlier) \tag{8}$$

Where:

λ : denotes the threshold value.
$minOutlier$: is the minimum outlier value in a dataset.

The edge computing process of the outlier detection is summarized as follows:

1) Each sensor node S_i periodically collects environmental data and offloads it to a nearby edge server. The process of collecting and transmitting sensor data to edge servers is represented in Algorithm 1.
2) Each edge server runs one copy of the LSTM prediction model and performs two tasks: (i) predicts the subsequent sensor data measurement using the LSTM prediction model, and (ii) performs outlier detection. When a data point is received, it is sent to the predictor module to forecast its next timestamp value. The predicted and actual values are then passed to the outlier detector. Then the following check is made: if the difference between the predicted value and the actual value exceeds a certain threshold, this value will be considered an outlier. Once the detection is done, the original data and the detection results are sent to the cloud for further analysis. Algorithm 2 describes in detail the outlier detection process.

Algorithm 1 Collection-Propagation

Begin
- V : a set of vectors containing captured values from different sources $\{V_0, V_1, ..., V_n\}$
- $Texp$: waiting time to receive data from sources
- Aut : Level of battery in percent
- $Thrld$: Threshold or critical value of Level of battery
-Init (Texp) : initialise the waiting time value;
-To-Edge-Server (V) : send the vector V to the corresponding edge server ;
repeat
  Init (Texp);
  repeat
    Receive-Data(); */*Run the sub process : sensing and receiving data */*
    Insert the received value into Vi ; */*Run the sub process : sensing and receiving data */*
  until (Texp expired)
  To-Edge-Server (V);
until (Aut $\leq Thrld$)
End.

Algorithm 2 Edge-Analyse

Begin
- Sni : a set of received values from sensor node i
- λ : Threshold value
- $Count - received$: Vector to store the number of received data from each sensor-node initialised to zero
- $LSTM(V)$: a method to predict the next sensor data measurement based on current vector V
- $Buffer$: a set of vectors which store the last predicted values by LSTM module of each sensor node i
- Si : Sensor node i
- $Alert(Si)$: detection of an outlier measurement from sensor node i
while (True) **do**
  wait until received sensing vector data Vi, from a sensor node i;
  increments Count-received [i] ;
  if (Count-received [i] != 1) && (Buffer[i] - Vi $\geq \lambda$) **then**
    Alert(Si);
  end if
  Buffer [i] = LSTM(Vi);
  Send-cloud (Vi); */*Send the vector Vi to the cloud for a global detection */*
end while
End.

3) Upon receiving data from the edge servers, the cloud server stores this data on online database servers. Afterward, it analyzes this data and makes decisions accordingly.

V. Experimental Results and Analysis

In this section, we have used our proposed method to detect outliers in temperature, humidity, and voltage data. First, we designed three LSTM-based predicting models named Temp-Forecasting model, Hum-Forecasting model, and Volt-Foreasting model. Then, Outliers are detected based on the results obtained from these predicting models. This study is accomplished using Google colaboratory (Colab). The implemented code is written in Python version 3.7.14, and the three LSTM-based prediction models were built using TensorFlow 2.8.2 and Keras 2.8.0. Pandas, NumPy, and matplotlib are some of the libraries used for data analysis in this work.

A. Dataset Description

The Intel Berkeley Research Lab (IBRL) dataset [22] is a publicly available real-world dataset released by IBRL labs. This dataset consists of sensor measurements gathered from 54 Mica2Dot sensor nodes deployed in IBRL between February 28th and April 5th, 2004. The sensed data was collected once every 31 seconds using the TinyDB in-network query processing system. This dataset contains 2.3 million rows and eight columns. The columns' names and explanations are presented in Table II. The two columns labeled Date and Time were combined into a single column called Date_Time. Epoch is a sequence number that increases monotonically with each mote, Moteids range from 1 to 54, the temperature is expressed in degrees Celsius (°C), the humidity is temperature-corrected relative humidity and ranges from 0 to 100%, Light is in Lux, and voltage is in volts.

In the experiments, we selected the temperature, humidity, and voltage measurements recorded from 28-03-2004 to 05-03-2004 (12000 log rows). Since this data is not annotated and lack labeled information, data preprocessing is required before creating the forecasting models.

To evaluate the performance of our proposed approach, we have generated twelve synthetic datasets by injecting outliers into the original dataset. Outliers are injected into the original dataset by sampling a certain percentage of data measurements and randomly flipping their values.

B. Performance Metrics

To assess the performance of our proposed method, we select three evaluation metrics: accuracy (ACC), F1-score, and False Alarm Rate (FAR). They are calculated as follows:

$$ACC = \frac{TP + TN}{TP + FP + TN + FN} \tag{9}$$

$$F1\_Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{10}$$

Where :

$$Precision = \frac{TP}{TP + FP} \tag{11}$$

$$Recall = \frac{TP}{TP + FN} \tag{12}$$

$$FAR = \frac{FP}{FP + TN} \tag{13}$$

Where TP (True Positive) denotes the number of correctly detected outliers, FP (False Positive) is the number of normal

TABLE II
DESCRIPTION OF THE VARIABLES IN THE IBRL DATASET.

Date	Time	Epoch	*Mote_id*	Temperature	Humidity	Light	*Voltage*
yyyy-mm-dd	hh:mm:ss.xxx	int	int	real	real	real	real

TABLE III
LSTM-BASED PREDICTION MODELS SETTINGS.

Model	Dependant/independent variables	Hideen Layers	Nodes per h_layers	*Dropout*	Batch size	*Epoch*	Learning rate	*Optimizer*
Temp_Forecasting Model	Humidity,voltage/ temperature	1	75	0.2	5	1000	0.0001	Adam
Hum_Forecasting Model	Temperature, voltage/ humidity	1	50	0.2	32	2000	0.0001	Adam
Volt_Forecasting Model	Temperature, humidity/ voltage	1	50	0.2	32	500	0.0001	Adam

measurements that were incorrectly identified as outliers, TN (True Negative) is the number of correctly identified normal measurements, and FN (False Negative) denotes the number of outliers that were incorrectly identified as normal.

C. Numerical Results and Discussion

1) Prediction models: In this study, we have used 80% of the dataset to train the three LSTM-based prediction models and the remaining 20% to test their performances. These models are trained using only normal data (without outliers).

We have conducted numerous experiments to find the optimal hyper-parameters for the three models. The hyper-parameters considered for these three models are batch size, epoch, learning rate, and optimization algorithm. The Mean Squared Error (MSE) was employed as the loss function. Table III lists the optimal hyper-parameters used for training the LSTM-based prediction models. The comparisons between the predicted and actual values of the temperature, humidity, and voltage are shown in Figure 3. We can see from the figure that the predicted values of the temperature, humidity, and voltage almost coincide with the actual values, which validates our forecasting models and allow us to proceed to the next phase, the outlier detection.

The Root Mean Squared Error (RMSE), given in (14), has been used to evaluate the predictive performances of the three predicting models. The RMSE should have a value that is as low as possible. Table IV shows the predictive performances of the three LSTM-based forecasting models. The results show that the Temp Forecasting model reported the forecasted temperature with an RMSE of 0.068, and the Hum-Forecasting model predicted the relative humidity with an RMSE of 0.113. The Volt-Forecasting model provides the expected voltage with an RMSE of 0.006.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{\left(\hat{y}_i - y_i\right)^2}{n}} \tag{14}$$

TABLE IV
PREDICTION MODELS PERFORMANCE'S RESULTS.

Model	RMSE(Training)	RMSE(Test)
Temp_Forecasting Model	0.084	0.068
Hum_Forecasting Model	0.164	0.131
Volt_Forecasting Model	0.008	0.006

2) Outliers detection model: In the last experiment, we aimed to ensure that the proposal could detect outliers with a higher percentage. As detailed in the section IV, once the LSTM predictor model forecasts the next sensor data measurement, the detection of outliers can be launched at each time step. In fact, we have computed the accuracy (ACC), the F1 score and the FAR rate for each model namely: temperature, humidity and voltage model. We have also added outlier values for each dataset (5%, 10%, 15% and 20%) in order to evaluate the robustness of our proposal. The tables IV, V and VI show that the obtained results can be considered as good. We can easly notice that the worst result is 94.5% for the accuracy when we added 20% outlier values to the voltage dataset. As for the other metrics (F1_score and FAR), the rates are acceptable. This can be explained by the fact that when we chose outliers to add to the datasets, this process caused confusion with other existing outliers in the same datasets.

TABLE V
PERFORMANCE RESULTS OF TEMPERATURE OUTLIER DETECTION.

Datasets	ACC(%)	F1_Score(%)	FAR(%)
Temp_with_5%of_outliers	98.4	86	1.3
Temp_with_10%of_outliers	97	86	2.4
Temp_with_15%of_outliers	96.3	85.9	2.8
Temp_with_20%of_outliers	95	84.4	4

TABLE VI
PERFORMANCE RESULTS OF HUMIDITY OUTLIER DETECTION.

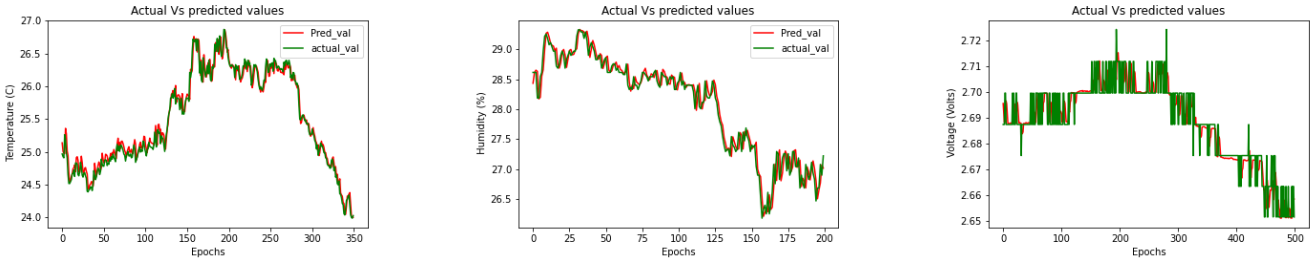
Datasets	ACC(%)	F1_Score(%)	FAR(%)
Hum_with_5%_of_outliers	98.5	87	1.3
Hum_with_10%_of_outliers	97	86	2.4
Hum_with_15%_of_outliers	96.1	86	2.7
Hum_with_20%_of_outliers	95.8	85	2.9

TABLE VII
PERFORMANCE RESULTS OF VOLTAGE OUTLIER DETECTION.

Datasets	ACC(%)	F1_Score(%)	FAR(%)
Volt_with_5%_of_outliers	97	79	2.3
Volt_with_10%_of_outliers	96.6	85	3
Volt_with_15%_of_outliers	95	84.5	4
Volt_with_20%_of_outliers	94.5	85.7	5

VI. CONCLUSION

This paper proposes a novel deep learning-based outlier detection method for IoT sensor data in hierarchical edge



(a) Actual Vs Predicted values of Temperature

(b) Actual Vs Predicted values of Humidity

(c) Actual Vs Predicted values of voltage

Fig. 3. Comparison of Predicted and Actual Values of temperature, humidity, and voltage.

computing. The proposed scheme consists of two modules: an LSTM-based predictor, which predicts the subsequent sensor data measurement, and an outlier detector, which decides whether a data point is an outlier. Experimental results reveal the efficiency of our proposed method for outlier detection in sensor data. We plan in future work to improve the present paper by adding a mechanism for finding the best edge and cloud servers to run our predictor and outlier detection models.

REFERENCES

[1] K. Ashton *et al.*, "That 'internet of things' thing," *RFID journal*, vol. 22, no. 7, pp. 97–114, 2009.

[2] V. R. Shinde, P. P. Tasgaonkar, and R. Garg, "Environment monitoring system through internet of things (iot)," in *2018 International Conference on Information, Communication, Engineering and Technology (ICICET)*. IEEE, 2018, pp. 1–4.

[3] F. Hu, D. Xie, and S. Shen, "On the application of the internet of things in the field of medical and health care," in *2013 IEEE international conference on green computing and communications and IEEE Internet of Things and IEEE cyber, physical and social computing*. IEEE, 2013, pp. 2053–2058.

[4] X.-G. Luo, H.-B. Zhang, Z.-L. Zhang, Y. Yu, and K. Li, "A new framework of intelligent public transportation system based on the internet of things," *IEEE Access*, vol. 7, pp. 55 290–55 304, 2019.

[5] F. Zhang, "Research on applications of internet of things in agriculture," in *Informatics and Management Science VI*. Springer, 2013, pp. 69–75.

[6] S. Xi, C. Zhang, Z. Cai, and Y. Xu, "Cost control and project cost analysis of intelligent building under internet of things," *Mobile Information Systems*, vol. 2021, 2021.

[7] J. Jiang and K. Su, "Management platform architecture of modern tobacco logistics based on internet of things technologies," in *LISS 2012*. Springer, 2013, pp. 1403–1409.

[8] H.-N. Dai, H. Wang, G. Xu, J. Wan, and M. Imran, "Big data analytics for manufacturing internet of things: opportunities, challenges and enabling technologies," *Enterprise Information Systems*, vol. 14, no. 9-10, pp. 1279–1303, 2020.

[9] M. A. Jamshed, K. Ali, Q. H. Abbasi, M. A. Imran, and M. Ur-Rehman, "Challenges, applications and future of wireless sensors in internet of things: A review," *IEEE Sensors Journal*, 2022.

[10] M. G. R. Alam, M. M. Hassan, M. Z. Uddin, A. Almogren, and G. Fortino, "Autonomic computation offloading in mobile edge for iot applications," *Future Generation Computer Systems*, vol. 90, pp. 149–157, 2019.

[11] V. Barnett and T. Lewis, "Outliers in statistical data," *Wiley Series in Probability and Mathematical Statistics. Applied Probability and Statistics*, 1984.

[12] S. Sadik and L. Gruenwald, "Online outlier detection for data streams," in *Proceedings of the 15th Symposium on International Database Engineering & Applications*, 2011, pp. 88–96.

[13] C. Titouna, F. Naït-Abdesselam, and A. Khokhar, "Dods: A distributed outlier detection scheme for wireless sensor networks," *Computer Networks*, vol. 161, pp. 93–101, 2019.

[14] C. Titouna, F. Nait-Abdesselam, and A. Khokhar, "A multivariate outlier detection algorithm for wireless sensor networks," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6.

[15] B. Chander and G. Kumaravelan, "Outlier detection strategies for wsns: A survey," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 8, Part B, pp. 5684–5707, 2022.

[16] A. Ayadi, O. Ghorbel, A. M. Obeid, and M. Abid, "Outlier detection approaches for wireless sensor networks: A survey," *Computer Networks*, vol. 129, pp. 319–333, 2017.

[17] T. Luo and S. G. Nagarajan, "Distributed anomaly detection using autoencoder neural networks in wsn for iot," in *2018 ieee international conference on communications (icc)*. IEEE, 2018, pp. 1–6.

[18] K. Zhang, K. Yang, S. Li, D. Jing, and H.-B. Chen, "Ann-based outlier detection for wireless sensor networks in smart buildings," *IEEE Access*, vol. 7, pp. 95 987–95 997, 2019.

[19] B. Chander and K. Gopalakrishnan, "Auto-encoder—lstm-based outlier detection method for wsns," in *Machine Learning, Deep Learning and Computational Intelligence for Wireless Communication*. Springer, 2021, pp. 109–119.

[20] Q. Zhang, M. Ye, and X. Deng, "A novel anomaly detection method for multimodal wsn data flow via a dynamic graph neural network," *Connection Science*, vol. 34, no. 1, pp. 1609–1637, 2022.

[21] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[22] "Intel lab data home page," http://db.csail.mit.edu/labdata/labdata.html, March, 2004.