Information Sciences 553 (2021) 49–65

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

A hypergrid based adaptive learning method for detecting data faults in wireless sensor networks

Lingqiang Chen [a], Guanghui Li [a,*], Guangyan Huang [b]

[a] School of Artificial Intelligence and Computer Science, Jiangnan University, Wuxi 214122, China
[b] School of Information Technology, Deakin University, Melbourne 3125, Australia

ARTICLE INFO

Article history:
Received 2 June 2020
Received in revised form 27 October 2020
Accepted 7 December 2020
Available online 16 December 2020

Keywords:
Wireless sensor networks
Data faults/anomalies
Hypergrid
Lazy learning
Continuous learning

ABSTRACT

In wireless sensor networks (WSNs), data anomalies/faults often occur due to the limited resources and unreliability of sensor nodes. Many traditional anomaly detection methods are designed in a batch manner, but for the nature of streaming data in WSNs, continuous anomaly detection method is preferred. Existing methods often detect only a single type of faults but cannot detect multiple types of faults that actually are more common in the sensor data. Therefore, this paper provides a Hypergrid based Adaptive Detection of Faults (HADF) method, which adopts hypergrid and statistical analysis to recognize three types of faults in the sensor data, including outliers, stuck-at faults, and noisy faults. HADF is a distributed method running on sensor nodes, which can reduce the influence of concept drift in unstable streaming data through combining both lazy leaning and continuous learning to adaptively update its normal profile. In the experimental study, we have manually inserted different types of faults into two real-world datasets, and the results demonstrate that HADF achieves higher accuracy with reasonable efficiency for detecting the data faults than four counterpart methods.

1. Introduction

With the rapid development of wireless sensor network technologies, Internet of Things (IoT) applications have covered a variety of fields [32], such as industrial control [3,13], forest environmental monitoring [20,22] and ocean environment monitoring [9], network security monitoring, healthcare and smart medical [2,29], and precision agriculture [1,37]. According to Cisco statistics, there will be 14.7 billion connected devices in IoT by 2023 [8]. Each sensor network application may contain hundreds even thousands of devices that continuously generate data streams and accumulate them into a huge amount.

However, sensor nodes may generate abnormal data due to the limitation of the node resources and complex deployment environment. Three typical categories of data faults are stuck-at faults, noisy faults, and outliers [23]. First, when sensor nodes are in low power or out of power, they will lose the capability of normal monitoring or data gathering, and thus the output values of the sensors will keep abnormally constant; we call them stuck-at faults. Second, sensors may produce intermittent or persistent noisy faults due to electromagnetic interference. Third, sensors often sample data with sharp changes, called outliers. Also, because of their short duration, it is difficult to pinpoint the cause. The poor-quality data with the above mentioned faults will increase extra communication overhead to the sensor network, and even affect decision

* Corresponding author.
E-mail address: ghli@jiangnan.edu.cn (G. Li).

https://doi.org/10.1016/j.ins.2020.12.011
0020-0255/

making for those data-based applications. For example, stuck-at faults will result in a poor performance for the time series prediction methods. Thus, effective fault detection is critical for improving the quality of sensor data and monitoring.

In recent years, diverse anomaly detection methods using different principles have been developed for WSNs, such as statistics-based, density-based, distance-based, cluster-based, learning-based methods [6], and trust based methods [15,19,27]. Many detection methods often work under different assumptions or definitions of the normal data and may be effective for detecting one type of data faults. According to the statistical result in [32], the outlier is the most commonly addressed fault type in literature, and some papers also focus on noisy or stuck-at faults.

In this paper, we propose an effective Hypergrid based Adaptive Detection of Faults (HADF) method for detecting the abovementioned three typical types of sensor data faults. In HADF method, two different detectors (hypergrid based and statistical analysis based detectors) are used for detecting multiple types of faults. Also, HADF adopts both lazy learning and continuous learning methods. Moreover, we redesign a robust L1 detection region (RDR) by adding a transition region to the traditional L1 detection region (L1-DR).

HADF applies a hypergrid structure to obtain the normal profile of recent data. The existing methods most related to ours are hypergrid based detection methods, such as HGDB and HypGridE [10,35]. HGDB is a basic online hypergrid based method and updates its NP in a lazy learning manner. HypGridE is an ensemble method that forms a strong detector by combining several HGDB detectors. Our scheme is different from the above two methods in three aspects:

- HADF combines not only the basic HGDB detector but also the statistical analysis based detector.
- A novel L1 detection region is proposed, which is different from the improved L1 detection region in [10,35].
- HADF updates its NP in two manners: lazy learning and continuous learning.

Thus, this paper has the following three contributions. In each contribution we also point out the advantage of our proposed scheme.

- We propose an effective method for detecting multiple types of sensor data faults (i.e., outliers, stuck-at faults, and noisy faults). Two different detectors applied in our schemes improve the accuracy for detecting various types of faults.
- We adopt both lazy learning and continuous learning methods to increase the adaptability of the normal profile for mitigating concept drift. It can not only improve the detection rate but also reduce the communication overhead caused by frequent updates.
- We provide a robust L1 detection region for improving the accuracy of existing improved L1-DR in the proposed hypergrid based method. Like other improved L1-DR, RDR has lower computational complexity than L1-DR. Besides, HADF applied RDR owns higher robustness to the data at the center or boundary than those using other improved L1-DR in [10,35].

The rest of this paper is structured as follows. In Section 2, the related work is reviewed and summarized. The important preliminaries are presented in Section 3. We detail the proposed HADF method in Section 4. Section 5 demonstrates the proposed method in extensive experiments. Finally, we conclude the paper in Section 6.

2. Related work

Many existing methods have already been used to detect data faults in wireless sensor networks [7,24,25,33,38]. These methods can be divided into two types: centralized and distributed, based on their architectures [5].

The centralized method is usually performed on the base station with all data collected from sensor nodes. Base stations generally have abundant resources to store large amounts of historical data to train complex models. In [12], a centralized anomaly detection method is applied to the medical application. The physiological parameters of the monitoring object are collected by the remote nodes and uploaded to the base station. Subsequently, a sequential minimum optimization regression method is applied to predict current data using historical data and marks the anomaly data by comparing the difference between predicted data and sampled data with a predefined threshold. Many other common anomaly detection algorithms, such as CNN [34], LSTM [16], auto-encoder [11], can also be designed as the centralized methods for WSNs. However, the centralized methods often result in high communication cost, since they need to obtain all sensor data and additional information, such as the source of the data, that may be required to pinpoint the cause of the fault. In sensor networks, the communication energy is much higher than that of computing. Thus, the centralized approaches may not be suitable for detecting faults occurring at the node.

Fortunately, many distributed methods have been used to overcome the limitations of the centralized methods. Here, tasks are divided into subtasks among nodes. The communication overhead of networks can be reduced, since the sensor nodes (SNs) only need to transfer processed data or model parameters to their neighbors or cluster head (CH). A distributed histogram-based method is presented in [36]. Each SN sets the local histogram with the global value range. Then, CH forms a histogram of global data by merging all the local histograms from SNs. After getting the global histogram parameters from CH, SN performs anomaly detection locally. Similar to the method proposed in [36], a distributed method in [26] is also divided into two stages: training stage and test stage. In the training stage, SN gathers the normal observations to form a

local normal model (LNM) and sends it to the CH for constructing the global normal model (GNM). In the test stage, SN tests sampled measurements using the GNM. Another version of distributed detection methods is that SNs form detection models by using the parameters from their neighbors without the coordination of CH. In [39], an ellipsoidal support vector machine (SVM) is used to detect outliers, which reduces the computational overhead by fixing the center of hyper-ellipsoid at the origin. Each node trains its local hyper-ellipsoid SVM and shares the model parameters with neighbor nodes. Consequently, nodes form a global model using the shared parameters and detect the data faults using both local and global models.

HGDB [35] and HypGridE [10] are also designed in a distributed manner. HGDB establishes a hypergrid structure in feature space that consists of hypercubes with fixed size and maps training data onto the structure to record the distribution of data. All SNs in one cluster share their local distribution of data with CH so that CH can draw the global normal profile (NP). Sequentially, SN performs online anomaly detection locally with the NP and the threshold received from CH. To meet the dynamic change of surroundings, HGDB relearns its NP periodically. HypGridE constructs a strong detector by using several basic HGDB models. Moreover, HypGridE decreases the number of searching adjacent hypercubes using an improved L1 detection region that considers the data mapped to the boundary.

However, the aforementioned distributed methods are relatively simple because of their limited training samples. Therefore, when the concept drift occurs, the distribution of data changes and causes the failure of the NP (i.e., the NP is out of date). To solve this problem, HADF adopts the combination of continuous learning and lazy learning to finely tune the NP in each SN. In this way, we can not only reduce the communication overhead of frequent updates due to the NP failure but also improve the adaptability of NP.

3. Preliminaries

In this section, we introduce preliminary knowledge, including the types of faults and the hypergrid based anomaly detection method. All the abbreviated symbols are listed in Table 1.

3.1. Anomalies in WSN streaming data

In the real-world applications, the sensor nodes cannot avoid collecting erroneous data. Anomalies/data faults are classified into three categories mentioned before: outlier faults, stuck-at faults and noisy faults [23], as shown in Fig. 1. Due to the lack of the ground truth of sensor data and the short duration of outlier fault, we cannot analyze the cause accurately. According to [30], lack of ground-truth values, the fault refers to a deviation from the expected value. This kind of anomaly was first investigated in [4], as an additive outlier for time series data. In [28], it was pointed out that additive outliers are easy to be identified, because these outliers do not influence other data in context. When the node is affected by some electromagnetic interference or other factors, the collected data will be contaminated by noise. Additionally, when the remaining energy of nodes fails to support the normal operation of the sensor, or the sensor hardware fails, it will collect continuous constant values, and these data are different from the normal data. The specific definitions of anomalies in streaming data are as follows:

Outlier Faults: Outlier faults are those points that are significantly different from other data sampled in the near time. Formally, outlier fault can be expressed as:

$$y(t)' = y(t) + \Delta, \tag{1}$$

where $y(t)$ is the true value collected at time t, Δ is an error value, and $y(t)\prime$ is the abnormal data.

Stuck-at Faults: A node collects data that remain at the constant value η for a long time. Formally, Stuck-at faults can be expressed as Eq. 2, where x is a timestamp ($x \in [t, t+n]$), and stuck-at faults last for n timestamps.

$$\{y(x)' = \eta \,|\, x \in [t, t+n]\} \tag{2}$$

Noisy Faults: When the data are contaminated by noise, the variance of the data will increase. In [23], it assumes that noisy data obeys a new normal distribution, different from that of normal data. Let μ and σ be the mean and standard deviation of normal data. We set the abnormal data distribution by adding or multiplying a small constant value (γ, σ) to the original mean and variance, where $N(\mu, \sigma)$ is a random normal distribution function. Formally, noisy fault can be expressed as:

Table 1
Description of the abbreviated symbols.

Abbreviations/symbols	Description
HADF	Hypergrid based adaptive detection of faults method
RDR	Robust L1 detection region
CH	Cluster head node
SN	Sensor node
NP	Global normal profile in one cluster that records all data distribution in hypergrid structure
NP*	Local normal profile that records the distribution of data of each SN in hypergrid structure

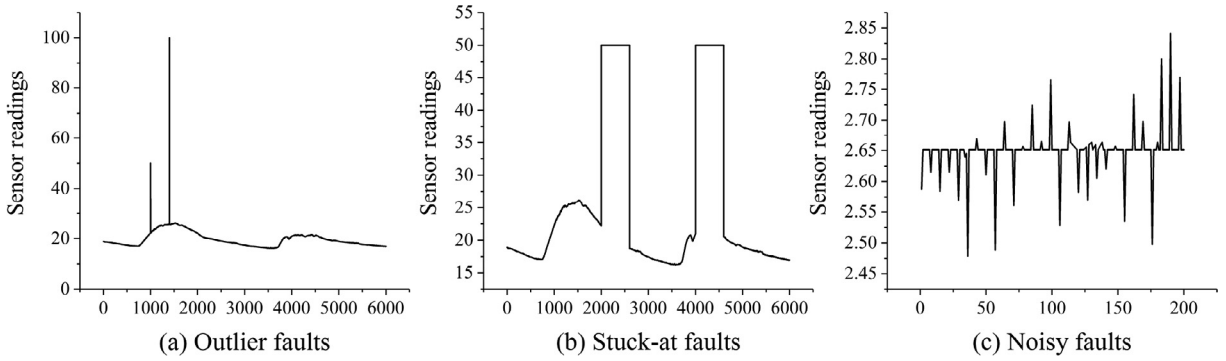

Fig. 1. Three fault types in sensor readings.

$$y(t)' \sim N(\mu + \gamma, \lambda * \sigma). \tag{3}$$

3.2. Hypergrid based anomaly detection method

Hypergrid structure composes continuous hypercubes. The side length and diagonal length of the hypercube are h and d, respectively, where $d = h\sqrt{q}$, and q is the dimensions of feature space. Hypergrid based algorithm is a distance-based method that detects the anomalies according to the data size in the detection region of test data. If the data size is larger than the threshold, the test data is marked as normal, otherwise not. Therefore, before detecting, the distribution of data should be obtained in the learning procedure. A distributed method is used to get a global normal profile [10,35] as follows:

- SN in one cluster shares the summary of its local data (like linear sum, linear sum of squares, maximum and minimum values) with its CH.
- CH summaries the global data information (like mean, standard deviation, global minimum) and forms a new hypergrid structure.
- SN normalizes its local data and maps them onto the hypergrid structure to form a local normal profile NP*, which contains each non-repeating position of the non-empty hypercube and corresponding data size in them. Besides, SN estimates a local threshold K^* using its local data.
- CH merges all NP* and K^* from SNs, and forms the global NP and the threshold K.
- Finally, SN performs online detection locally, according to NP and K received from CH.

The memory size of NP is increased by the number of attributes in each data vector. Therefore, an encoding technique is utilized to compress the position of each data into a fixed-length sequence of bits, irrespective of its dimension [35]. For convenience, the test data discussed below are all normalized.

The detection process of hypergrid based methods can be roughly divided into two parts: the mapping process and the counting process. The mapping process is performed as below. If there is a multi-dementional data object $x_t = [x_{t_1}, \ldots, x_{t_q}]$ sampled at time t, where x_t is a q-dimensional vector, then x_t can be mapped onto a hypercube $C_{u_1,\ldots,u_q}$ according to Eq. (4). The coefficient, c, is a constant value that is set as $c > |\min|$, by which the entire feature space will be shifted into positive coordinate spaces, and the positive integers, $u_1, \ldots, u_q$, denote the indices of a hypercube.

$$u_i = \left\lfloor \frac{x_i + c}{h} \right\rfloor, \quad i = 1, \ldots, q. \tag{4}$$

Before counting the data size in the neighbor of the test sample, we introduce the concept of the detection region.

In hypergrid structure, there are two types of detection regions (DR), called Layer-1 neighbors of a hypercube (L1) and Layer-2 neighbors of a hypercube (L2). L1 and L2 of $C_{u_1,\ldots,u_q}$ can be represented as Eqs. (5) and (6), respectively.

$$\mathrm{L1}(C_{u_1,\ldots,u_q}) = \{C_{d_1,\ldots,d_q} | \mathbf{d} = \mathbf{u} \pm 1, C_{d_1,\ldots,d_q} \neq C_{u_1,\ldots,u_q}\} \tag{5}$$

$$\mathrm{L2}(C_{u_1,\ldots,u_q}) = \{C_{d_1,\ldots,d_q} | \mathbf{d} = \mathbf{u} \pm 2, C_{d_1,\ldots,d_q} \neq C_{u_1,\ldots,u_q}, C_{d_1,\ldots,d_q} \notin \mathrm{L1}(C_{u_1,\ldots,u_q})\}. \tag{6}$$

Fig. 2 shows a 2D example of hypergrid structure. Given a threshold, K, of the data size, we can get the following property [17].

Property 1. If there are less than K data in $C_{u_1,\ldots,u_q} \cup \mathrm{L1} \cup \mathrm{L2}$, all test data in $C_{u_1,\ldots,u_q}$ are anomalous.

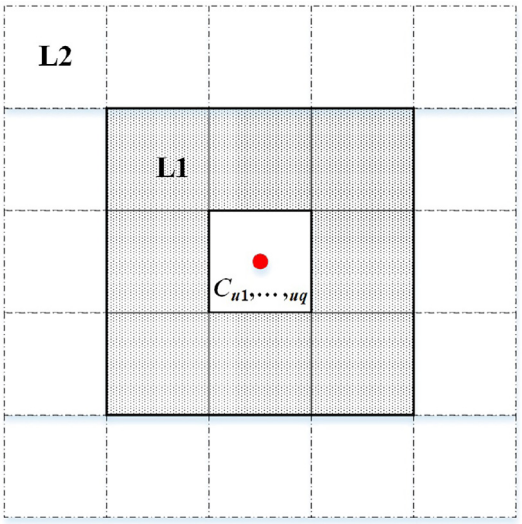

Fig. 2. A 2D example of hypergrid structure.

The data size in L1 and L2 of mapping hypercube of test data would be calculated by the hypergrid based method. According to Property 1, counting will stop for a certain hypercube when the cumulative number is larger than the threshold, K, and the test data is verified as normal. Nevertheless, the size of the detection region of hypercube increases exponentially with the number of attributes increases. A simplified property for anomaly detection is proposed in [35] to reduce the computational complexity.

Property 2. If there are less than K data in $\mathrm{C}_{u_1,\ldots,u_q} \cup \mathrm{L1}$, all test data in $C_{u_1,\ldots,u_q}$ is anomalous.

In [35], a substituted detection region is applied to further decrease the detection region. We assume that a test data object $\boldsymbol{x} = [x_1, \ldots, x_q]$ is mapped into hypercube $C_{u_1,\ldots,u_q}$, its substituted detection region R can be represented as follows:

$$\begin{aligned} \mathrm{R}(x) &= \{\mathrm{C}_{d_1,\ldots,d_q} \,|\, d_i = u_i, u_i + e_i, i = 1, \ldots, q\} \\ e_i &= \begin{cases} +1, & x_i - \lfloor x_i \rfloor > 0.5 \\ -1, & x_i - \lfloor x_i \rfloor \leqslant 0.5 \end{cases} \end{aligned} \tag{7}$$

Another version of the improved L1 detection region is proposed in [10]. Considering the data mapped onto the boundary, the new detection region R is defined as follows:

$$\begin{aligned} \mathrm{R}(x) &= \{\mathrm{C}_{d_1,\ldots,d_q} \,|\, d_i = u_i, u_i + e_i, i = 1, \ldots, q\} \\ e_i &= \begin{cases} +1, & x_i - \lfloor x_i \rfloor > 0.5 \\ -1, & x_i - \lfloor x_i \rfloor < 0.5 \\ \{-1, +1\}, & x_i - \lfloor x_i \rfloor = 0.5 \end{cases} \end{aligned} \tag{8}$$

In practice, it is hard to confirm a proper substituted DR [35] of data mapped in the transition region, and the number of data points mapped to the center or edge of the hypercube is significantly less than other regions [10]. Therefore, we redefine the L1-detection region (L1-DR) by a more robust one (Abbrev. RDR). The detailed RDR of $\boldsymbol{x} = [x_1, \ldots, x_q]$ that maps into a hypercube $C_{u_1,\ldots,u_q}$ can be represented as Eq. (9), where w is the width of the transition region, and $w \in [0, 1]$.

$$\begin{aligned} \mathrm{RDR}(x) &= \{\mathrm{C}_{d_1,\ldots,d_q} \,|\, di = u_i, u_i + e_i, i = 1, \ldots, q\} \\ e_i &= \begin{cases} +1, & x_i - \lfloor x_i \rfloor > 0.5 + w/2 \\ -1, & x_i - \lfloor x_i \rfloor < 0.5 - \mathrm{w}/2. \\ \{-1, +1\}, & 0.5 - w/2 \leqslant x_i - \lfloor x_i \rfloor \leqslant 0.5 + w/2 \end{cases} \end{aligned} \tag{9}$$

According to Eq. (9), it is obvious that the size of RDR is influenced by w. Without loss of generality, we assume that the distribution density of data points in the hypercube is the same. Let the side length of hypercube be 1, and its dimension be q. The volume of the hypercube is derived as follows:

$$\begin{aligned} V &= [2(0.5 - w/2) + w]^q = 1 \\ &= \sum_{k=0}^{q} C_q^k [2(0.5 - w/2)]^k w^{q-k} \end{aligned} \tag{10}$$

According to Eq. (9–10), the average size of RDR is derived as follows:

$$E(q, w) = \sum_{k=0}^{q} C_q^k [2(0.5 - w/2) * 2]^k (w * 3)^{q-k} \tag{11}$$

$$\begin{aligned} E(3, w) &= \sum_{k=0}^{3} C_3^k [2(0.5 - w/2) * 2]^k (w * 3)^{3-k} \\ &= w^3 + 6w^2 + 12w + 8 \end{aligned} \tag{12}$$

4. The hypergrid based adaptive detection of faults method

In this section, we first introduce the overview of our scheme and then detail HADF in three parts: distributed learning process, detection/continuous learning process, and complexity analysis.

In WSNs, many sensor nodes are deployed in the real environment and always formed cluster structures [31]. In this study, we focus on applying anomaly detection on sensor nodes in each cluster. Consider a hierarchical wireless sensor network deployed in a geographic area of interest, we give a background diagram of HADF as shown in Fig. 4. The WSN consists

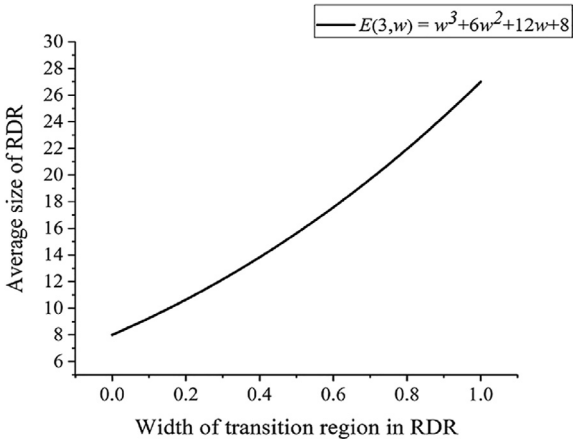

Fig. 3. Average size of RDR of the 3D hypergrid structure.

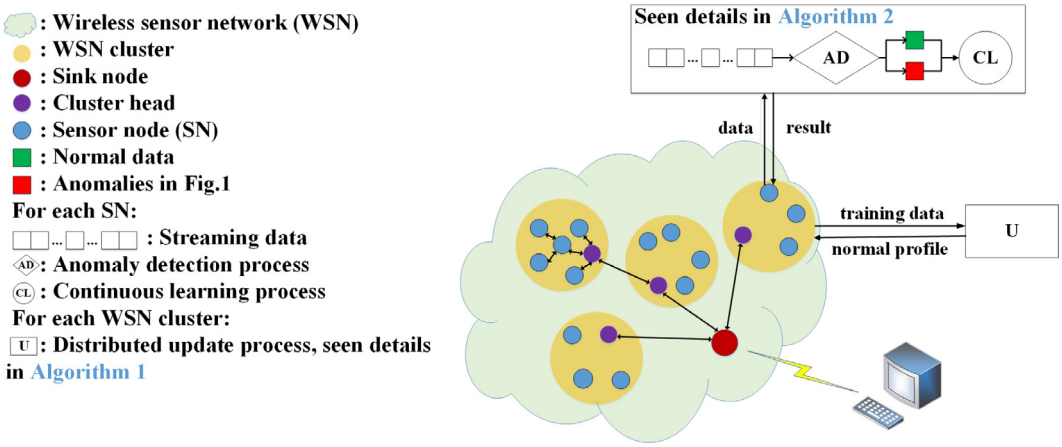

Fig. 4. Overview of HADF.

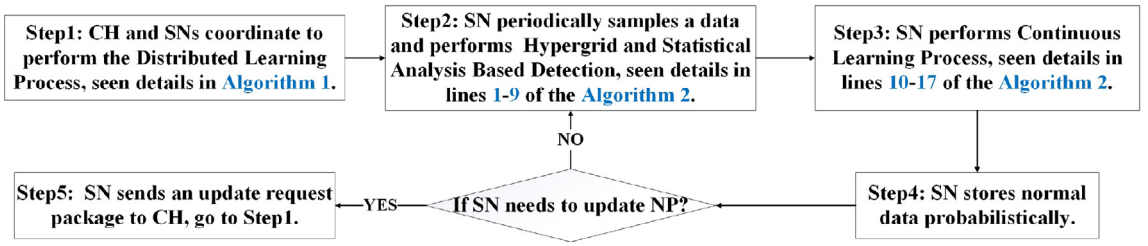

Fig. 5. Workflow of HADF.

of one sink node and several clusters. In each cluster, there are one CH and several SNs. HADF can be deployed in each cluster with the coordination of CH and SNs as shown in Fig. 5. First, CH and SNs cooperate to complete the distributed learning process, i.e., the lazy learning process within the cluster. Then, SNs perform hypergrid and statistical analysis based detectors on periodically sampled data using the updated NP. After detection, SNs apply continuous learning to add the information of new data into NP. Besides, normal data will be stored with probability in the local memory. SNs within one cluster repeat step 2 to step 4 until one of SNs needs to update NP and sends the update request to the CH. Finally, a new round of the distributed learning process begins. To release the impact of the data loss on the update, we can apply two underlying mechanisms: data recollection and packet retransmission mechanisms.

4.1. Distributed learning process

The distributed learning process of HADF in SN will be triggered when one of the following conditions is satisfied.

- The amount of data stored locally exceeds the predefined value.
- Data in the local buffer are all replaced by new data.

During the distributed learning process, some basic parameters of hypergrid structure are updated, among which the side length, h, and the threshold, K, determine the performance sensitively. However, it is hard to obtain an optimal h, due to lack of prior knowledge. In [35], an estimation interval of h is proposed by minimizing the mean integrated squared error of the detection region. The interval of h can be calculated by Eq. (13), where q and n are the dimensions of feature space and mapping data size, respectively.

$$h \in \left[\left(\left(\frac{3}{8} \right)^q \frac{6q}{Z} \right)^{\frac{1}{q+2}}, \left(\left(\frac{1}{2} \right)^q \frac{6q}{Z} \right)^{\frac{1}{q-2}} \right] \tag{13}$$

$$Z = n \sum_{i=1}^{q} \left(2^{q+1} \pi^{\frac{q}{2}} \right)^{-1} \tag{14}$$

The threshold, K, is a standard to describe whether a test data is in a dense region or sparse. If data size N in the RDR of test data is less than K, the data is abnormal. In [35], K can be approximately estimated by calculating the mean data size in the detection region of training data. Let $|\mathrm{RDR}(x_i)|$ be the number of data in the RDR of x_i; the mean data size can be estimated as Eq. (15). The rate, r, is applied to make the probability density function in hypergrid structure continued over the entire range [35], where max_z and min_z represent the maximal and minimal values of the training data in attribute z, respectively. The parameter, h, is set as the maximum value in the distribution interval in Eq. (13). In a distributed manner, each SN works out K^*, and CH obtains the final K by averaging all of the K^* from SNs.

$$K^* = \frac{\mathrm{r}}{S} \sum_{i=1}^{s} |\mathrm{RDR}(\mathrm{x}_i)| \tag{15}$$

$$r = \frac{|NP^*|}{\prod_{z=1}^{q} \lceil \frac{\max_z - \min_z}{h} \rceil} \tag{16}$$

$$K = \frac{1}{m} \sum_{i=1}^{m} K_i^* \tag{17}$$

The update of NP is similar to that in [35], which is coordinated by SNs and CH. The process of update is shown in Algorithm 1.

Algorithm 1. Distributed learning process.

1: **while** SN triggers update do
2: SN asks CH for update
3: CH broadcasts update command among its Cluster
4: Each SN sends its local statistical information to CH
5: CH gets a global statistical information, forms a hypergrid structure, and sends these parameters back to SNs
6: Each SN normalizes its local data, and gets local NP* and K^*
7: CH merges all NP* and K^*, and forms global NP and threshold K
8: **end while**

4.2. Detection and continuous learning process

After receiving the parameters of HADF from CH, each SN samples data periodically and performs detection according to the new NP and K. The detection process includes two detectors: the hypergrid based detector and the statistical analysis based detector. We assume that the ith SN (SN_i) samples a data vector $x_t = [x_{t_1}, \ldots, x_{t_i}, \ldots, x_{t_q}]$ at time t, which is normalized as $y_t = [y_{t_1}, \ldots, y_{t_i}, \ldots, y_{t_q}]$. SN_i maps the data onto a hypercube $C_{u_1,\ldots,u_q}$. The mean and standard deviation of the latest n data are $\boldsymbol{\mu} = [\mu_1, \ldots, \mu_i, \ldots, \mu_q]$ and $\boldsymbol{\sigma} = [\sigma_1, \ldots, \sigma_i, \ldots, \sigma_q]$, respectively. Subsequently, data is tested by two detectors. The property of hypergrid based detector is defined as follows.

Property 3. If there are more than K data in RDR of test data, the data is marked as normal by hypergrid based detector, otherwise the data is marked as anomalous.

In practice, sensor readings have a precision error, $\boldsymbol{p} = [p_1, \ldots, p_i \ldots, p_q]$, which can be found in sensor datasheet. To eliminate the influence of accuracy, the statistical analysis based detector calculates the absolute difference, d_i, between x_{t_i} and μ_i, and compares the difference, d_i, with the accuracy, p_i. Besides, d_i is also compared with $3\sigma_i$. The detailed property is given as follows.

Property 4. If the absolute difference, d_i, is less than p_i or $3\sigma_i$, the data is marked as normal by the statistical detector, otherwise the data is marked as anomalous.

Combining the two detectors, we give the property of HADF as follows.

Property 5. Combining the different results of hypergrid based detector and statistical analysis based detector, r_1 and r_2, the situations of test data, $\boldsymbol{x}$, are diverse.

- If both r_1 and r_2 are true, then $\boldsymbol{x}$ is normal.
- If both r_1 and r_2 are false, then $\boldsymbol{x}$ is obviously abnormal.
- If r_1 is true, but r_2 is false, then $\boldsymbol{x}$ is marked as a contextual anomaly, which is detected as normal in the global data distribution but is a local fault.
- If r_2 is true, but r_1 is false, then concept drift has occurred if this situation continuously happens.

Those obvious anomalies in Fig. 1 (such as outlier faults and stuck-at faults) can be easily detected by hypergrid based detector because they are far different from the normal data. In the meantime, the statistical analysis based detector can be a lightweight method to detect the contextual and noisy anomalies.

During detection, we count the data size (S) in RDR of test data and record the amount of data $\left(\widehat{S}\right)$ in RDR except the hypercube, where the test data is mapped. Once the value of S is larger than K, counting stops. The continuous learning process is influenced by $\widehat{S}$, the result of the statistical analysis based detector (r_2) and NP. Generally, for the data marked as normal by statistical analysis detector, if its mapped information is not recorded in NP, or its $\widehat{S}$ is larger, it will be recorded into the NP with a higher weight. Although some low-intensity noisy data may be recorded by NP, their initial weights are small and have little influence on the detection. The use of $\widehat{S}$ as the size of neighbor data, not S, is to reduce the likelihood that continuous constant anomalies will be learned. Besides, we increase a probabilistic mechanism to further reduce the possibility of anomalies being recorded. Finally, NP will be completely replaced at the next update.

Algorithm 2 details the procedure of detection and continuous learning process.

Algorithm 2. Hypergrid and statistical analysis based detection and continous learning process.

Input:
Normal profile, $NP = [pos, weight]$
The threshold of hypergrid based method, K
The precision error of sensor reading, $\boldsymbol{p} = \left[p_1, \ldots, p_i \ldots, p_q\right]$

1: **repeat**
2: SN samples a data x_t
3: Normalizes test data x_t
4: maps x_t onto hypergrid structure $C_{u_1,\ldots,u_q}$ according to Eq. (4)
5: counts S and $\widehat{S}$ in RDR of $C_{u_1,\ldots,u_q}$ according to Eq. (9)
6: $r_1 \Leftarrow$ gets Hypergrid based detector result according to **Property 3**
7: obtains mean μ and standard deviation σ of recent normal data
8: $r_2 \Leftarrow$ gets Statistical analysis based detector result according to **Property 4**
9: *result* $\Leftarrow$ gets final detection result according to **Property 5**
10: **if** triggers continuous learning process then
11: **if** $C_{u_1,\ldots,u_q} \in$ NP then
12: NP $\Leftarrow$ increases the weight of $C_{u_1,\ldots,u_q}$ in NP according to r_2 and $\widehat{S}$
13: **else**
14: initializes the weight w of $C_{u_1,\ldots,u_q}$ according to r_2 and $\widehat{S}$
15: NP $\Leftarrow$ adds $\left[C_{u_1,\ldots,u_q}, w\right]$ into NP
16: **end if**
17: **end if**
18: **until** Next distributed learning process in Algorithm 1

4.3. Complexity analysis

The main computation overhead of HADF is spent on hypergrid and statistical analysis detectors. In the update process of hypergrid based detector, a main complexity of $O(nlogn)$ is incurred at each SN for sorting all data coordinates (n) and forming NP*. Meanwhile, CH requires a complexity of $O(ulogu)$ to sort all NP*, where $u = \sum_{i=1}^{m} |NP_i^*|$. The detection process of hypergrid based detector is incurred at SNs and yields a complexity of at least $O(log\, v)$ and at most $O\left(3^{q-1} log\, v\right)$, where v is the size of NP $(v < u \ll n)$, and q is the dimension of feature space. Additionally, HADF performs the statistical analysis detector at SNs by comparing the measurement with statistical metrics (like mean, standard deviation) in each attribute, which requires a complexity of $O(q), q$ is the dimension of feature space. Overall, HADF has a complexity of $O(nlogn)$.

5. Experimental results

To evaluate our proposed HADF, we have conducted several experiments on two datasets. All the algorithms were implemented using Python scripts, on a PC with a 2.6 GHz Intel(R) Core(TM) i7-1075H CPU, 16G memory, and the Windows10 operating system.

5.1. Datasets

The real-world datasets include Intel Berkeley Research Lab datasets [18] and SensorScope datasets [14,32], as shown in Table 2.

The Intel Lab dataset was collected from 54 Mica2Dot sensors deployed in the Intel Berkeley Research Lab (IBRL) between February 28th and April 5th, 2004. Each node collects humidity, temperature, light, and voltage values once every 31 s.

Table 2
Experimental datasets.

Datasets	Data Type	Node ID	Anomaly rate (%)	Size
IBRL	Temperature, Humidity & Light	21, 22, 26	5	20,000
SensorScope (SS)	Ambient temperature, Surface temperature & Humidity	12, 15, 17	5	20,000

SensorScope is an outdoor sensor network deployment project consisting of several deployments. Here we use the Lausanne Urban Canopy Experiment (LUCE) deployment on the EPFL campus since July 2006. LUCE consists of 97 weather-stations with sensors for sensing the environmental attributes once every 30 s such as ambient temperature, surface temperature and relative humidity.

Most datasets in WSN are unlabeled and the cost of manually labelling a large amount of data is high. Also, few datasets contain various types of anomalies to verify our methods. Therefore, some anomaly insertion methods have been applied to address this issue [30,35]. In this paper, we also adopt anomaly insertion methods to verify our method. We adopt the same assumption as in [6,30,32] that only a few sensors fail at the same time. First, we select the data interval with no obvious anomalies in the original datasets. Then, in each trial of the experiments, the same as in [30,35], we randomly chose proportional indices (5%) that contain the anomalous data. The samples at the indices are replaced by abnormal data generated by certain methods. Due to the large variation (from 0 to 1,800) of light data in the IBRL dataset, the anomaly insertion methods for light data are different from other types of data. The Δ of outlier faults in Eq. (1) obey a random distribution of (10, 50) ((1,000, 1,500) for light data). Stuck-at faults are simulated by continuous insertion of a constant value as shown in Eq. (2), where the duration is set as 20 samples and the value of η is chosen from [90, 100] ([900, 1,000] for light data). Let μ and σ be the mean and standard deviation of original data respectively. A random normal distribution function, $N(\mu+\gamma, \lambda*\sigma)$, is used to generate noisy faults. In this paper, we set $\gamma = 2$ and $\lambda = 1.5$ ($\gamma = 100$ and $\lambda = 1.5$ for light data).

Considering multivariate anomalies, we further subdivide the types of anomalies according to the number of attributes that are influenced by a certain fault. For example, Outlier(1) represents that one of the attributes is contaminated by outlier fault at one moment, Noise(2) represents that two of the attributes are contaminated by noisy faults at one moment, and Constant(3) represents that all three attributes are contaminated by stuck-at faults at one moment.

5.2. Accuracy metrics

For the two-class classification problem, the samples can be classified into four types: true positive (*TP*), true negative (*TN*), false positive (*FP*) and false negative (*FN*). TP represents the volume of anomalies correctly labeled as abnormal. TN represents the volume of normal data correctly labeled as normal. FP denotes the volume of normal data wrongly labeled as abnormal. And FN denotes the volume of anomalies wrongly labeled as normal. The following metrics are used to evaluate the performance of HADF: *FPR*, Precision (*P*), Recall (*R*) and F_1-score (F_1).

$$FPR = \frac{FP}{TN + FP}. \tag{18}$$

$$P = \frac{TP}{TP + FP}. \tag{19}$$

$$R = \frac{TP}{TP + FN}. \tag{20}$$

$$F_1 = 2 * \frac{P * R}{P + R}. \tag{21}$$

Considering both runtime and F_1-score, we define a new metric ψ called comparative score as below:

$$\psi = \frac{F_1}{time} * 100. \tag{22}$$

The higher value of ψ represents the better performance of the method.

5.3. Performance under different widths

According to Eq. (12), the average size of RDR increases exponentially with the size of *w*. We have adopted two optimization strategies to reduce the computational complexity of hypergrid based detector:

- When the data is judged to be normal, the search will stop.
- The hypercube of the data map has the highest search priority.

For the nature of sensor data, adjacent data always have similar values. Therefore, those adjacent data are prone to be mapped onto the same hypercube whose weight increases with the addition of normal data during the continuous learning process. Once the weight is larger than the threshold, subsequent test data that mapped into the same hypercube will be labeled as normal directly.

To obtain the appropriate value of *w*, we have studied the influence of different *w* values on both the detection performance and the average searched hypercubes. IBRL is used in this experiment, and its data volume is 9,000, among which the first 3,000 samples are pre-training data. The rest 6,000 data are test data that contain 300 outlier faults. Buffer size

of the node is set to 3,000, and normal data is saved with a probability of 0.8. As shown in Fig. 3, the increase in width expands the size of the detection region, which improves the detection accuracy at the expense of computing time. First, we have taken 21 values from 0 to 1 by an interval of 0.05. From the result in Fig. 6(a), F_1-score is increased with the width w and reaches about 94.5% when w is between 0.8 and 1. Then, we have taken 21 values from 0.8 to 1 using an interval of 0.01. The value of the F_1-score reaches the maximum when w is 0.87 in Fig. 6(b). Due to the continuous learning and detection mechanisms, the average number of the searched hypercubes is smaller than the average size of RDR. After comprehensive consideration, we set the value of w as 0.87 in the following experiments.

5.4. Performance for detecting multiple types of data faults

In this subsection, we demonstrate the performance of HADF for detecting multiple anomalies. HGDB [35], HypGridE [10] and KitNET [21] are used as counterpart methods. As shown in Fig. 7(a)–(c), HGDB is a basic distributed hypergird based method, which is used as a benchmark, and HypGridE is an ensemble learning method that combines multiple HGDB detectors to form a strong one by weighting all the results. The difference between HADF and the above two methods in detection is that HADF applies not only a hypergrid based detector but also a statistical analysis detector. For the learning process as shown in Fig. 7(d)–(f), both HGDB and HypGridE update their normal profile periodically. However, SN using HADF updates its NP in two manners (the distributed learning process in the cluster and the continuous learning process at local). Additionally, we also compare HADF with a neural network method called KitNET. KitNET is an unsupervised, integration-based anomaly detection method, and it uses an ensemble of neural networks called autoencoders to collectively differentiate between normal and abnormal data. To evaluate the efficiency of continuous learning, we have improved the continuous learning mechanism to form a new HGDB scheme and develop a new HGDBI method. The difference between HADF and HGDBI is whether to use RDR.

All of the data in Table 2 are used for experiments, among which the first 10,000 samples of each node are training sets, and the rest are test data that contains 500 anomalies (5%). KitNET has trained its model using all of the training data. The buffer sizes of sensors in the four hypergrid based methods are set as 3,000; therefore they only use the last 3,000 data of training data in each node to perform pre-training and relearn its model during detection. To ensure the reliability of the results, each group of experiments has been repeated for 5 times, and we have taken the average value as the final result.

As shown in Figs. 8–13, the performance of each algorithm varies with the different types and sizes of anomalies. The performance of the algorithm is increased with the increasing size of anomalies, except in a few cases for detecting the stuck-at faults by HGDB and HypGridE, as shown in Figs. 9 and 12(a). The recall of the four hypergrid based methods for outlier is higher than those of stuck-at faults and noisy faults. Due to the use of continuous learning method, HGDBI has higher precision than HGDB. Moreover, comparing the performance of HADF and HGDBI, RDR used in HADF has higher robust than improved L1 detection region used in HGDB. KitNET is sensitive to the number of anomalies, and its recall increases with the increasing number of anomalies.

In Figs. 8–13(a), overall, the recall of five methods is increased with the increasing of the number of faults, since the greater the number of anomalies, the measurement is farther away from the normal data. However, in Figs. 9 and 12(a), the performance of HGDB and HypGridE for stuck-at faults is decreased with the increase of the size of faults. Because most online anomaly detections require to dynamically update their normal profile. Therefore, the historical data may be stored locally with probability for the next update. Unlike outlier and noisy faults, the stuck-at faults show continuous constant values. As the number of stuck-at faults increases, their size that is stored randomly also increases, so it may lead to the formation of a "normal" cluster in the training data and eventually affect the performance of the following models. Additionally, we have tried to store only the data detected as normal, but both HGDB and HypGridE show a low precision. Storing only the normal data would obliterate a large number of misjudged data, which result in an incomplete normal profile, and get a worse result. HADF achieves a high precision because it adopts the strategy of preserving only normal data (Step 4 in Fig. 5)) to avoid the influence of stuck-at faults anomalies on the new normal profile. KitNET is sensitive to the number of faults. As the value and number of faults increase, the detection performance of KitNET changes significantly (except for Figs. 10 and 13(a)). The value of stuck-at faults is greater than the other two types of faults, so they are easier to be detected by KitNET for their higher anomaly scores. In Figs. 10 and 13(a), KitNET achieves a low recall (about 0.04 for IBRL and 0.24 for SS) for detectings noisy faults when compared with other algorithms. Actually, there are many causes for the above results. Firstly, there are fewer attributes in the data set, resulting in a small number of encoders; Secondly, the value of the noisy faults is close to the data area of the training set, which results in a low anomaly score in the output layer. The statistical analysis detector adopted by HADF effectively improves the detection accuracy of noisy anomalies.

In Figs. 8–13(b), due to the concept drift of the sensor data, HGDB achieves a high FPR (about 37.3% for IBRL, 29.2% for SS), which results in a low precision. Although HypGridE has adopted ensemble HGDB detectors to form a strong one, it still has a non-negligible FPR (about 13.6% for IBRL, 10.2% for SS); meanwhile, its recall is reduced. To relieve the influence of concept drift, HGDBI and HADF adopt continuous learning method (Step 3 in Fig. 5) that effectively reduces FPR (HGDBI: about 19.1% for IBRL, 12.6% for SS; HADF: about 0.70% for IBRL, 0.74% for SS) and improves the precision. The precision of KitNet is also increased with the increasing of the number of faults. However, KitNet achieves lower precision than our scheme due to more false positives caused by a fixed threshold. Besides, Low anomaly scores for noisy faults also result in low precision of KitNet. Comparing the precision of HADF and HGDBI, the RDR used in HADF improves the robustness of our scheme and significantly increases the precision.

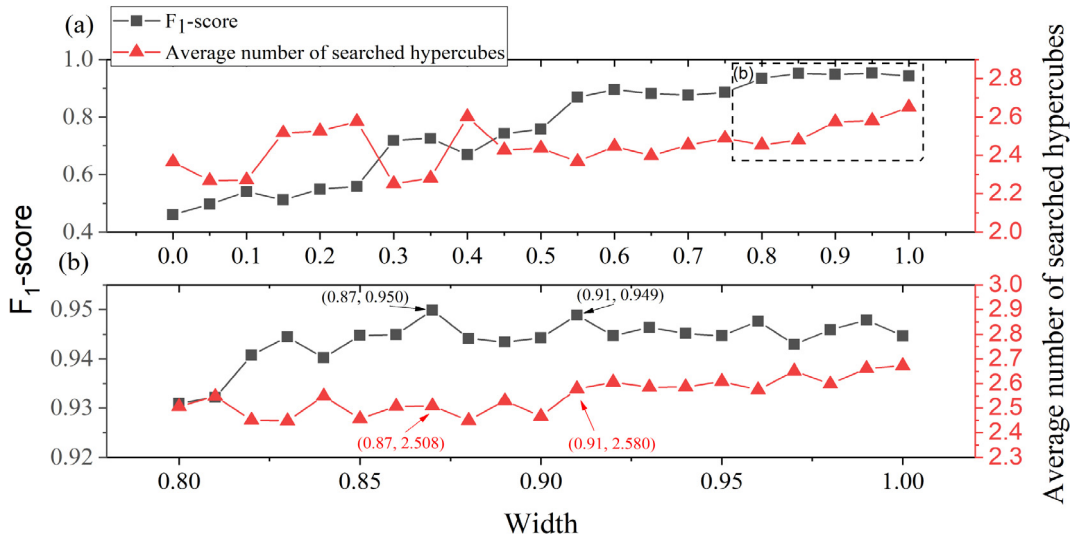

Fig. 6. Performance under different widths.

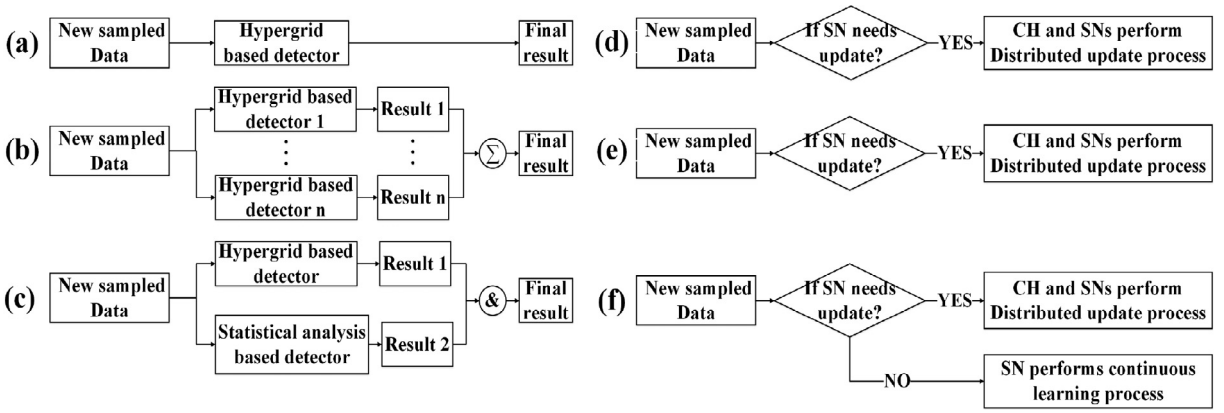

Fig. 7. The overview of three hypergrid based methods: (a), (b) and (c) are the detection processes of HGDB, HypGridE and HADF, respectively; (d), (e) and (f) are the learning processes of HGDB, HypGridE and HADF, respectively.

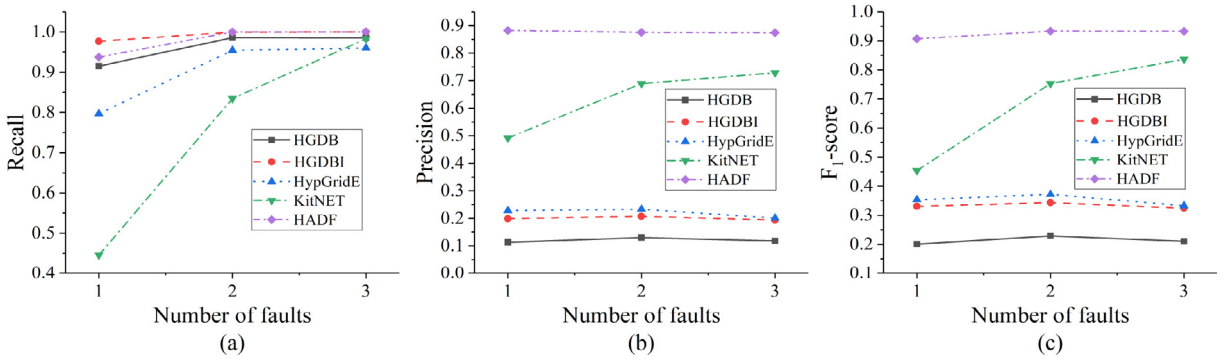

Fig. 8. Performance of detecting outlier faults in IBRL.

In Figs. 8–13(c), we compare the F_1-score of different methods in terms of recall and precision. In Figs. 8 and 11(c), HADB achieves the lowest score for detecting outliers and stuck-at faults due to its lowest precision. The scores of HGDB and HypGridE for stuck-at faults are decreased with the increasing number of faults, because of the formation of a false 'normal' clus-

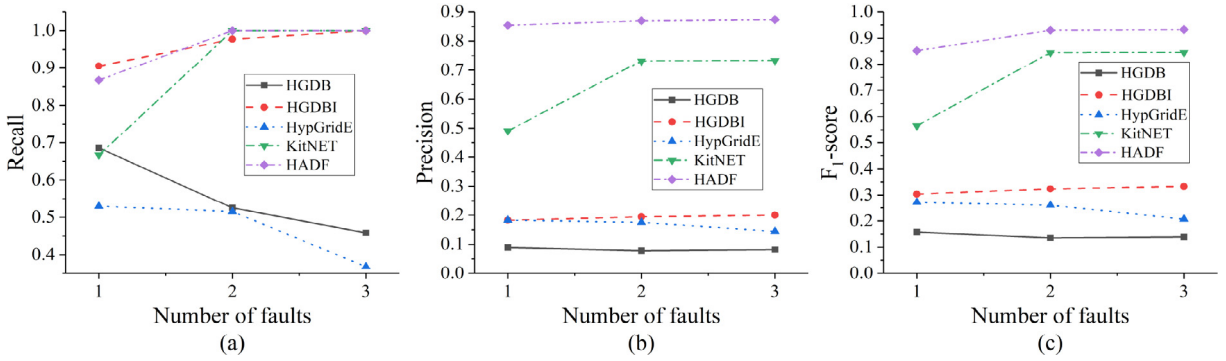

Fig. 9. Performance of detecting stuck-at faults in IBRL.

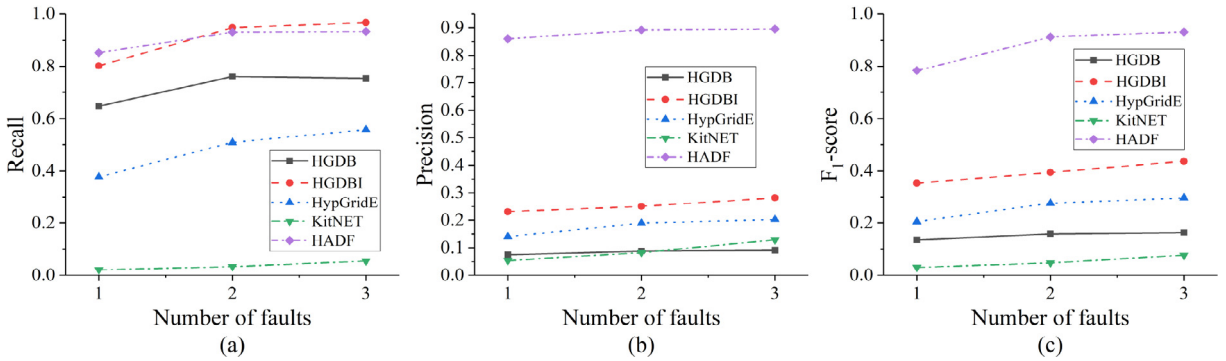

Fig. 10. Performance of detecting noisy faults in IBRL.

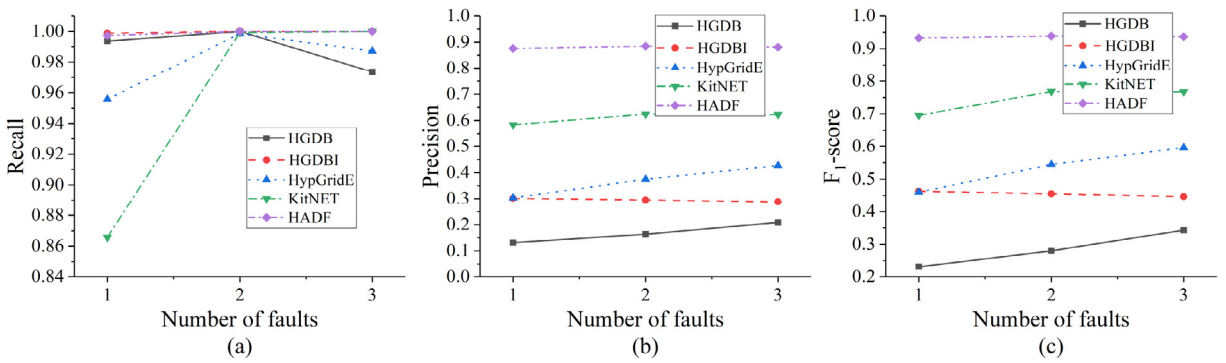

Fig. 11. Performance of detecting outlier faults in SS.

ter in the learning process, as mentioned before. The bad detection rate of KitNET for noisy faults leads to the worse score in Figs. 10 and 13(c). Overall, HADF outperforms other methods in the detection of various faults.

5.5. Scores of different algorithms

To obtain the scores of different algorithms using Eq. (22), we have recorded the average runtime of algorithms for different anomalies. HGDB, HGDBI, HypGridE and HADF update their NP and perform detection in a distributed fashion. Still, the execution of the algorithm in the simulation is single-threaded so that the runtime is divided by the number of nodes.

The runtime costs of five methods for IBRL and SS datasets are shown in Tables 3 and 4, respectively. We can see from Tables 3 and 4 that our proposed HADF and HGDBI spend a little more runtime than HGDB. However, the detection performance (considering all of precision, F1 score and recall) of our HADF as shown in Figs. 8–13 is obviously more accurate; HGDB is the fastest method but achieves the worst accuracy of fault detection. Our HADF consists of hypergrid detection

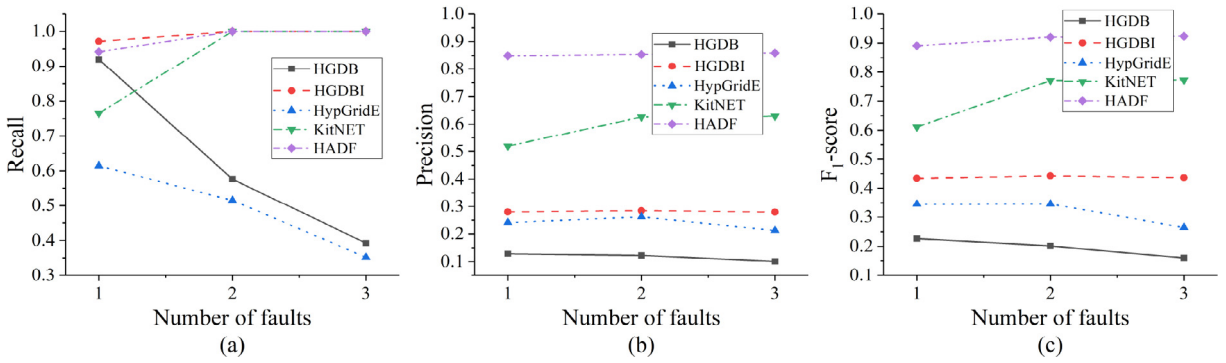

Fig. 12. Performance of detecting stuck-at faults in SS.

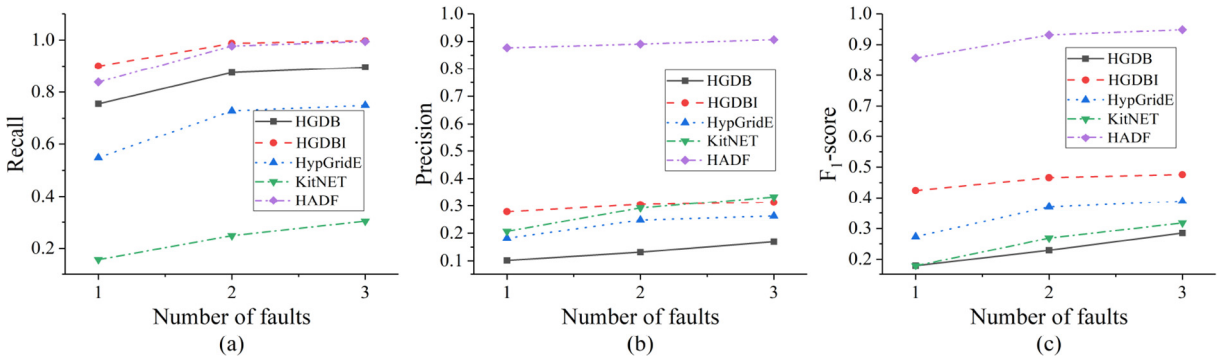

Fig. 13. Performance of detecting noisy faults in SS.

Table 3
Average runtime for fault detection in IBRL (unit: second).

Number of faults	HGDB	HGDBI	HypGridE	KitNET	HADF	
Outlier Fault	1	23.40	27.19	137.79	41.88	28.34
	2	23.29	27.36	139.34	42.14	28.24
	3	23.20	26.99	139.71	41.77	28.09
Stuck-at Faults	1	22.64	26.94	130.02	41.66	28.08
	2	22.47	27.04	123.96	41.78	28.16
	3	22.27	26.81	120.12	42.46	28.16
Noisy Faults	1	23.29	27.15	134.20	41.38	28.42
	2	23.51	27.40	137.18	41.26	28.56
	3	23.88	27.49	139.96	41.44	28.61

Table 4
Average runtime for fault detection in SS (Unit: second).

Number of faults	HGDB	HGDBI	HypGridE	KitNET	HADF	
Outlier Fault	1	24.49	28.78	135.51	42.22	29.28
	2	24.10	28.72	128.03	41.74	29.38
	3	23.77	28.55	122.36	41.74	29.34
Stuck-at Faults	1	23.64	28.58	127.90	41.70	29.43
	2	23.42	28.53	119.62	41.76	29.44
	3	23.56	28.41	113.18	41.83	29.35
Noisy Faults	1	25.22	28.52	137.37	42.13	29.80
	2	25.18	28.61	134.43	42.10	29.71
	3	24.97	28.84	130.26	41.75	29.84

process and statistical analysis step, so it spends a little more time than HGDB. By improving HGDB, our proposed HGDBI is used to evaluate the efficiency of the robust detection region (RDR). Using a RDR, HADF has a complexity at most $O\left(3^{q-1}log\,v\right)$ for each data detection, and HGDBI uses the improved detection region in [35], which has complexity at most

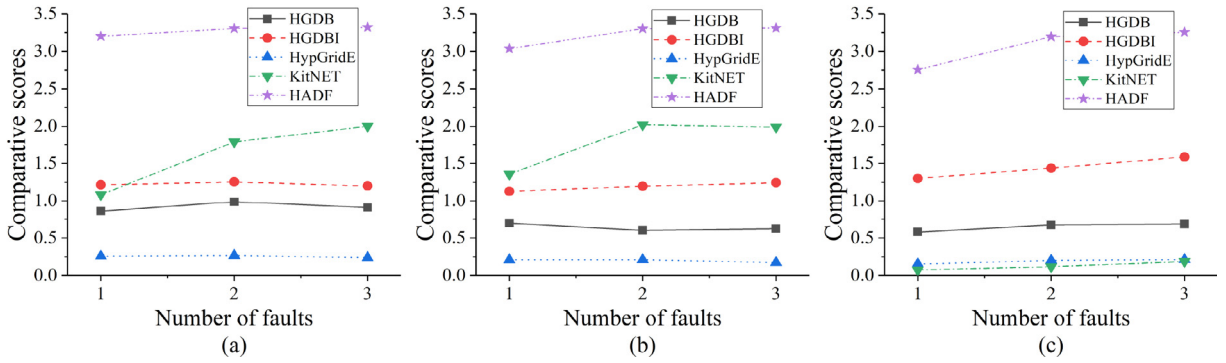

Fig. 14. Comparative scores for fault detection in IBRL (a: for Outlers b: for Stuck-at Faults c: for Noisy Faults).

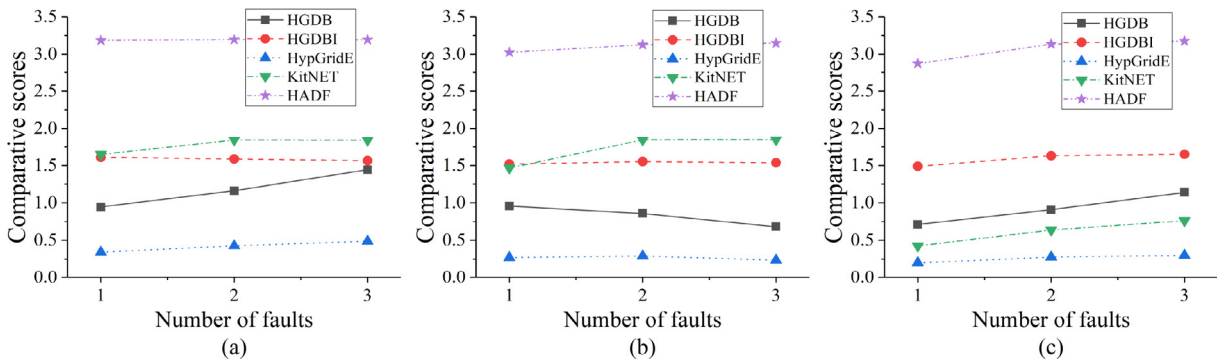

Fig. 15. Comparative scores for fault detection in SS (a: for Outlers b: for Stuck-at Faults c: for Noisy Faults).

$O\left(2^{q-1}log\,v\right)$ for each data detection. Meanwhile, the runtime of HADF is significantly less than HypGridE and KitNET. This is because HypGridE is an ensemble method, and its runtime increases linearly with the number of detectors, while KitNET needs massive data to train its auto-encoders.

Note that because of the same size of two datasets, the time costs in Table 4 are approximately the same as that in Table 3. Additionally, since the data distribution of SS is slightly sparser than that of IBRL, the four hypergrid based methods need to spend a little more time for searching the detection regions in NP.

We use comparative score (ψ) to evaluate the overall performance of the algorithms, and the results are shown in Figs. 14 and 15.

As shown in Fig. 14, the comparative scores of five methods for IBRL vary with the type and size of anomalies. Overall, the scores increase with the increasing number of faults, except in a few cases for detecting the stuck-at faults by HGDB and HypGridE. As mentioned before, stuck-at faults form a 'normal' cluster in the learning process, which causes a low precision for those faults in the following detection. From the results, although HGDB had the least running time in Table 3, its poor detection performance results in a low score (about 0.92 for outliers, 0.64 for stuck-at faults, and 0.65 for noisy faults). HypGridE takes several times more calculation time than other methods, so its comparative score is the worst/lowest (about 0.25 for outliers, 0.20 for stuck-at faults, and 0.19 for noisy faults). The detection accuracy of KitNET for noisy faults is low, which also leads to a bad result (about 0.13). HGDBI has a similar runtime with HADF, but its lower precision causes a lower score (about 1.22 for outliers, 1.19 for stuck-at faults, and 1.44 for noisy faults) than HADF. So, HADF outperforms other counterpart methods in detecting various anomalies.

The comparative scorces of five methods for SS are shown in Fig. 15. The results in Fig. 15 are similar with those in Fig. 14, and HADF also shows a better performance than other counterpart methods.

In summary, we can get three strategies from the experimental study to help achieve better performance for detecting multiple anomalies from WSNs. First, the distributed online anomaly detection methods for WSNs need to consider the possible concept drift of sensing data. Otherwise, it may result in low precision. Second, for some lazy-learning methods, if they adopt the strategy of random storage to store historical data, it is necessary to consider the impact of stuck-at faults. Especially for those distance-based detection methods, stuck-at faults may form a 'normal' cluster, which will result in low precision for this kind of anomalies. Third, the performance of the methods may vary with the change of the types and sizes of

anomalies. Therefore, we can obtain a comprehensive performance of the methods by verifying them for different types and sizes of anomalies that may occur in certain filed.

6. Conclusion

In this paper, a Hypergrid based Adaptive Detection of Faults (HADF) method for detecting multiple data faults in WSNs was proposed, which applied the hypergrid and statistical analysis based detectors to detect three types of data faults. Particularly, we redefined a robust L1 detection region for hypergrid based detector, so that HADF was more robust than those methods using the other two kinds of improved L1 detection regions. Furthermore, a continuous learning mechanism was adopted to reduce the influence of concept drift in sensor data. Extensive experiments on two real-world datasets had demonstrated that HADF performed better than three counterpart methods in terms of accuracy and comparative score (F_1-score/runtime).

CRediT authorship contribution statement

Lingqiang Chen: Conceptualization, Methodology, Validation, Writing - original draft, Visualization. **Guanghui Li:** Writing - review & editing, Supervision, Project administration, Funding acquisition. **Guangyan Huang:** Methodology, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (No. 62072216), Wuxi International Science and Technology Research and Development Cooperative Project (No. CZE02H1706), and the 111 Project (No. B12018).

References

[1] Pooyan Abouzar, David G Michelson, Maziyar Hamdi, Rssi-based distributed self-localization for wireless sensor networks used in precision agriculture, IEEE Trans. Wireless Commun. 15 (10) (2016) 6638–6650.
[2] Ahmad Alaiad, Lina Zhou, Patients' adoption of wsn-based smart home healthcare systems: an integrated model of facilitators and barriers, IEEE Trans. Professional Commun. 60 (1) (2017) 4–23.
[3] Juan Aponteluis, Juan Antonio Gomezgalan, F Gomezbravo, Manuel Sanchezraya, Javier Alcinaespigado, Pedro Teixidorovira, An efficient wireless sensor network for industrial monitoring and control, Sensors 18 (1) (2018) 182..
[4] Vic Barnett, Toby Lewis, Outliers in Statistical Data, third ed., Wiley Series in Probability and Mathematical Statistics, Wiley, Chichester, 1974.
[5] Satish Bhojannavar, Chetan Bulla, Vishal Danawade, Anomaly detection techniques for wireless sensor networks – a survey, Int. J. Adv. Res. Comput. Commun. Eng. 2 (10) (2013).
[6] Varun Chandola, Arindam Banerjee, Vipin Kumar, Anomaly detection: a survey, ACM Comput. Surveys 41 (3) (2009) 15.
[7] Poyu Chen, Shusen Yang, Julie A. Mccann, Distributed real-time anomaly detection in networked industrial sensing systems, IEEE Trans. Ind. Electron. 62 (6) (2015) 3832–3842.
[8] Cisco. Cisco Annual Internet Report, 2018–2023. White paper c11–741490. Cisco Systems Inc, San Jose, CA, 2020..
[9] X. Deng, Y. Jiang, L.T. Yang, L. Yi, J. Chen, Y. Liu, X. Li, Learning automata based confident information coverage barriers for smart ocean internet of things, IEEE Internet Things J. (2020) 1.
[10] Zhiguo Ding, Minrui Fei, Du. Dajun, Fan Yang, Streaming data anomaly detection method based on hyper-grid structure and online ensemble learning, Soft Comput. 21 (20) (2017) 5905–5917.
[11] Jinan Fan, Qianru Zhang, Jialei Zhu, Meng Zhang, Zhou Yang, Hanxiang Cao, Robust deep auto-encoding gaussian process regression for unsupervised anomaly detection, Neurocomputing 376 (2020) 180–190.
[12] Shah Ahsanul Haque, M.M. Rahman, Syed Mahfuzul Aziz, Sensor anomaly detection in wireless sensor networks for healthcare, Sensors 15 (4) (2015) 8764–8786.
[13] V.J. Hodge, S. O'Keefe, M. Weeks, A. Moulds, Wireless sensor networks for condition monitoring in the railway industry: a survey, IEEE Trans. Intell. Transp. Syst. 16 (3) (2015) 1088–1106.
[14] Francois Ingelrest, Guillermo Barrenetxea, Gunnar Schaefer, Martin Vetterli, Olivier Couach, Marc B. Parlange, Sensorscope: Application-specific sensor network for environmental monitoring, ACM Trans. Sensor Networks 6(2) (2010) 17..
[15] Bo Jiang, Guosheng Huang, Tian Wang, Jinsong Gui, Xiaoyu Zhu, Trust based energy efficient data collection with unmanned aerial vehicle in edge network, Trans. Emerg. Telecommun. Technol. (2020).
[16] Tae Young Kim, Sungbae Cho, Web traffic anomaly detection using c-lstm neural networks, Expert Syst. Appl. 106 (2018) 66–76..
[17] Edwin M. Knorr, Raymond T. Ng, Vladimir Tucakov, Distance-based outliers: algorithms and applications, VLDB J. 8 (3) (2000) 237–253.
[18] Dinesh Kumar Kotary, Satyasai Jagannath Nanda, Distributed robust data clustering in wireless sensor networks using diffusion moth flame optimization, Eng. Appl. Artif. Intell. 87 (2020) 103342..
[19] Ting Li, Wei Liu, Tian Wang, Zhao Ming, Xiong Li, Ming Ma, Trust data collections via vehicles joint with unmanned aerial vehicles in the smart internet of things, Trans. Emerg. Telecommun. Technol. (2020) e3956.
[20] Yanjun Li, Zhi Wang, Yeqiong Song, Wireless sensor network design for wildfire monitoring, in: 2006 6th World Congress on Intelligent Control and Automation, 1, 2006, pp. 109–113..
[21] Yisroel Mirsky, Tomer Doitshman, Yuval Elovici, Asaf Shabtai, Kitsune: an ensemble of autoencoders for online network intrusion detection. CoRR, abs/1802.09089, 2018..

[22] Antonio Molinapico, David Cuestafrau, Alvaro Araujo, Javier Alejandre, Alba Rozas, Forest monitoring and wildland early fire detection by a hierarchical wireless sensor network, J. Sensors 2016 (2016), 1–8.
[23] Arslan Munir, Joseph Antoon, Ann Gordonross, Modeling and analysis of fault detection and fault tolerance in wireless sensor networks, ACM Trans. Embed. Comput. Syst. 14 (1) (2015) 3.
[24] S. Rajasegarar, C. Leckie, J.C. Bezdek, M. Palaniswami, Centered hyperspherical and hyperellipsoidal one-class support vector machines for anomaly detection in sensor networks, IEEE Trans. Inf. Forensics Secur. 5 (3) (2010) 518–533.
[25] Sutharshan Rajasegarar, Christopher Leckie, Marimuthu Palaniswami, Hyperspherical cluster based distributed anomaly detection in wireless sensor networks, J. Parallel Distrib. Comput. 74 (1) (2014) 1833–1847.
[26] Murad A Rassam, Anazida Zainal, Mohd Aizaini Maarof, An efficient distributed anomaly detection model for wireless sensor networks, AASRI Procedia 5 (2013) 9–14.
[27] Yingying Ren, Zhiwen Zeng, Tian Wang, Shaobo Zhang, Guoming Zhi, A trust-based minimum cost and quality aware data collection scheme in p2p network, Peer-to-Peer Network. Appl. (2020).
[28] Shiblee Sadik, Le Gruenwald, Research issues in outlier detection for data streams, SIGKDD Explor. Newsl. 15 (1) (2014) 33–40.
[29] Osman Salem, Yaning Liu, Ahmed Mehaoua, Raouf Boutaba, Online anomaly detection in wireless body area networks for reliable healthcare monitoring, IEEE J. Biomed. Health Inf. 18 (5) (2014) 1541–1551.
[30] Abhishek B. Sharma, Leana Golubchik, Ramesh Govindan, Sensor faults: detection methods and prevalence in real-world datasets, ACM Trans. Sen. Netw. 6 (3) (2010) 23..
[31] Z. Tao, C. Wandong, L. Gang, Topology control for wireless sensor networks, in: 2007 IFIP International Conference on Network and Parallel Computing Workshops (NPC 2007), 2007, pp. 343–348..
[32] Hui Yie Teh, Andreas W. Kempa-Liehr, Kevin I. Kai Wang, Sensor data quality: a systematic review, J. Big Data 7 (1) (2020) 11..
[33] P. Vamsi, Anjali Chahuan, Machine learning based hybrid model for fault detection in wireless sensors data, Scalable Inf. Syst. 7(24) (2020) 161368..
[34] Franco Van Wyk, Yiyang Wang, Anahita Khojandi, Neda Masoud, Real-time sensor anomaly detection and identification in automated vehicles, IEEE Trans. Intell. Transp. Syst. 21 (3) (2020) 1264–1276.
[35] M. Xie, J. Hu, S. Han, H. Chen, Scalable hypergrid k-nn-based online anomaly detection in wireless sensor networks, IEEE Trans. Parallel Distrib. Syst. 24 (8) (2013) 1661–1670.
[36] M. Xie, J. Hu, B. Tian, Histogram-based online anomaly detection in hierarchical wireless sensor networks, in: 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications, 2012, pp. 751–759.
[37] S. Yoo, J. Kim, T. Kim, S. Ahn, J. Sung, D. Kim, A2s: automated agriculture system based on wsn, in: 2007 IEEE International Symposium on Consumer Electronics, IEEE, 2007, pp. 1–5.
[38] Y. Zhang, N.A.S. Hamm, N. Meratnia, A. Stein, M. van de Voort, P.J.M. Havinga, Statistics-based outlier detection for wireless sensor networks, Int. J. Geogr. Inf. Sci. 26 (8) (2012) 1373–1392.
[39] Yang Zhang, Nirvana Meratnia, Paul J.M. Havinga, Distributed online outlier detection in wireless sensor networks using ellipsoidal support vector machine, Ad Hoc Networks 11 (3) (2013) 1062–1074.