IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 28, 2019, accepted July 10, 2019, date of publication July 17, 2019, date of current version August 5, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2929581

An Isolation-Based Distributed Outlier Detection Framework Using Nearest Neighbor Ensembles for Wireless Sensor Networks

ZHONG-MIN WANG[1, 2], **GUO-HAO SONG**[1], **AND CONG GAO**[1, 2]
[1]School of Computer Science and Technology, Xi'an University of Posts and Telecommunications, Xi'an 710121, China
[2]Shaanxi Key Laboratory of Network Data Analysis and Intelligent Processing, Xi'an University of Posts and Telecommunications, Xi'an 710121, China

Corresponding author: Guo-Hao Song (sgh947662512@gmail.com)

This work was supported in part by the Shaanxi Science and Technology Coordination and Innovation Project under Grant 2016KTZDGY04-01, in part by the International Science and Technology Cooperation Program of the Science and Technology Department of Shaanxi Province, China, under Grant 2018KW-049, in part by the Special Scientific Research Program of Education Department of Shaanxi Province, China, under Grant 17JK0711, and in part by the Communication Soft Science Program of Ministry of Industry and Information Technology, China, under Grant 2019-R-29.

ABSTRACT In recent years, wireless sensor networks have been extensively deployed to collect various data. Due to the effect of harsh environments and the limitation of the computing and communication capabilities of sensor nodes, the quality and reliability of sensor data are affected by outliers. Thus, an effective outlier detection method is essential. The existing outlier detection methods have some drawbacks, such as extra resource consumption introduced by the size growth of a local detector, poor performance of combination methods of local detectors, and the weak adaptability of the dynamic changes of the environment, etc. We propose an isolation-based distributed outlier detection framework using nearest-neighbor ensembles (iNNE) to effectively detect outliers in wireless sensor networks. In our proposed framework, local detectors are constructed in each node by the iNNE algorithm. A new combination method taking advantage of the spatial correlation among sensor nodes for local detectors is presented. The method is based on the weighted voting idea. In addition, we introduce a sliding window to update local detectors, which enables the adaption of dynamic changes in the environment. The extensive experiments are conducted on two classic real sensor datasets. The experimental results show our framework significantly improves the detection accuracy and reduces the false alarm rate compared with other outlier detection frameworks.

INDEX TERMS Outlier detection, wireless sensor networks (WSN), iforest, local outlier factor (LOF), isolation using nearest neighbor ensembles (iNNE), sliding window.

I. INTRODUCTION

With the rapid development of microelectronics and wireless technology, Wireless Sensor Networks (WSNs) have been extensively widely applied to a large variety of fields, such as agriculture [1], healthcare [2], industry [3], [4], and smart home [5]. WSNs consist of a large number of sensor nodes densely deployed in the region of interest [6]. These sensing-embedded components are interconnected by wireless links and cooperate to collect high-fidelity data from different locations. The collected data is processed and transmitted to the sink node. The sink node is able to deliver the data to the end-user with communication devices (e.g. the Internet).

The associate editor coordinating the review of this manuscript and approving it for publication was Muhammad Khalil Afzal.

However, as the core component of the WSNs, sensor nodes are prone to producing outliers in the collected data [6]. The main reasons lie in three aspects [7].

1) *Resource limitation*. Sensor nodes have stringent resource constraints such as battery power, storage capacity, computing ability, and communication bandwidth. The limited resource and capability make the data generated by sensor nodes unreliable and inaccurate. Especially when battery power is exhausted, the probability of generating erroneous data will grow rapidly.
2) *Harsh environments*. Sensor nodes are often randomly deployed in an uncontrolled area. Thus, sensor nodes are suffering from interference and potential damage from a harsh environment [8].

3) *Malicious attacks*. An attacker can capture a sensor node and inject malicious codes to manipulate its operation. The captured node may produce erroneous data by the attacker's instructions.

The above internal and external issues may lead to unreliability of sensor data, which further influences quality of raw data and aggregated results. Since actual events occurred in the physical world, e.g., forest fire, earthquake or chemical spill, cannot be accurately detected using inaccurate and incomplete data, it is extremely important to ensure the reliability and accuracy of sensor data before the decision-making process. Due to the fact that outliers are one of the sources to greatly influence data quality, in this paper, we use outlier detection techniques for the data quality assurance in WSNs. Effective and efficient detection of outliers helps to improve the quality of the collected data which facilitates the decision-making of the interested events.

In WSNs, outlier detection, also known as anomaly detection, refers to the problem of finding measurements, which significantly deviate from the normal measurements in a specific time period. Various outlier detection frameworks for WSNs are reviewed in [10]: statistical-based [11]–[13], clustering-based [14]–[16], classification-based [17]–[19], spectral decomposition-based [20], [21], nearest neighbor-based [22], [23], and others [24]–[26]. Another classification based on the framework structure in [10] divides the outlier detection frameworks into centralized structures [14] and distributed structures [9], [17], [19], [26], [27]. In a centralized structure, the detection process is carried out at a central location, such as a sink node or a base station which aggregates the information from multiple nodes. In a distributed structure, the detection process runs at each node. In most cases, there is cooperation among a node and its neighbor nodes. For a centralized structure, the communication overhead of the data transmission to a sink node is very high. In WSNs, radio communication among nodes is the main reason for quick energy depletion. The energy consumption for transmitting one bit is more than processing thousands of bits in a node in WSNs [28]. Thus, a distributed structure that is able to enhance the detection efficiency and effectiveness is preferred [10].

There are lots of outlier detection frameworks based on the distributed structure [29]. However, there are still some drawbacks. Firstly, for a distributed structure, the detection algorithm constructs a local detector in each sensor node. Thus, the size growth of the local detector introduces a considerable extra computational burden and requires more memory in a sensor node. Meanwhile, in most distributed outlier detection frameworks, the information given by a local detector needs to be broadcasted in WSNs. Thus, the size growth of a local detector also causes additional communication overhead. Secondly, the procedures of the combination for local detectors in some distributed outlier detection frameworks are simple and not well described in the related paper. Though the final performance of frameworks is still good, the effect of combination methods is poor. Therefore, it has great potential to improve combination methods to make the final performance better. Thirdly, most of the existing distributed frameworks involve batch learning. Here, batch learning refers to that all the measurements are given to a local detector of the training phase for detector training. Thus, the detector cannot keep track of dynamic changes in the environment and becomes rigid over time.

To overcome the drawbacks of the distributed outlier detection frameworks mentioned above, an isolation-based distributed outlier detection framework using nearest neighbor ensembles (iNNE) for WSNs is proposed. Our framework contains a local detector and a global detector. The local detector is constructed based on the iNNE. The global detector is constructed by combining the local detectors of a node and its neighbor nodes, where the formation of a neighborhood is based on the spatial correlation among sensor nodes. The final detection of measurements is executed by a global detector of a sensor node. The main contributions of this paper are as follows:

1) An isolation-based distributed outlier detection framework using nearest neighbor ensembles is proposed. The framework combines the advantages of the isolation-based frameworks and the nearest neighbor-based frameworks. Our framework utilizes the idea of subset ensembles in the isolation-based frameworks to reduce computation burden and memory requirement. Moreover, we calculate outlier scores to identify outliers based on the distances between measurements, which is the idea of the nearest neighbor-based frameworks.
2) A new combination method for local detectors based on the weighted voting is introduced. The actual distances among sensor nodes are used as the weights. The benefit of our combination method is the broadcast of the information of local detectors in WSNs is avoided. Only the measurements and the positions of sensor nodes are exchanged within a neighborhood.
3) A self-adaptive algorithm is developed to update the proposed framework. The key idea of the algorithm is a sliding window that keeps track of the dynamic changes of sensor data. Especially, the algorithm eliminates the influence of the biased values caused by dynamic changes in sensor data. Thus, the accuracy of detection is improved.

The remainder of this paper is organized as follows. Section II reviews the existing outliers detection frameworks in WSNs and describes two famous algorithms in detail as preliminary knowledge. Section III presents an isolation-based distributed outlier detection framework using nearest neighbor ensembles for wireless sensor networks. Section IV evaluates the proposed framework with extensive experiments. Finally, conclusions and future work are presented in Section V.

II. RELATED WORK

In this section, the existing outlier detection techniques are summarized. In addition, the details of two typical algorithms which are used for comparison in section IV are provided as preliminary knowledge.

A. DETECTION TECHNIQUES

Outliers, or anomalies, are patterns in data that do not conform to a well-defined notion of normal behavior [30]. In this paper, we mainly discuss two types of outliers: discrete outliers and event outliers. Discrete outliers refer to the outliers that are distributed discretely and not correlated in time series. They are usually generated by noise and errors in WSNs. On the contrary, event outliers are distributed continuously in time series and are generated by events in WSNs. In general, event outliers can actually be described as a sequence of errors or erroneous readings in a streaming data set. The six categories of outlier detection models for WSNs given by [10] are as follows:

1) STATISTICAL-BASED

In [11], a Temporal Outlier Detection (TOD) algorithm based on time-series analysis and geostatistics is proposed. It utilizes an autoregressive model to predict the measurement at the next moment and build a confidence interval. If the actual measurement is not in the confidence interval, it is an outlier. However, the form of a normal time series may change over time. The update of the proposed model is not addressed. In [12], an online detection approach based on a Segmented Sequence Analysis (SSA) algorithm is proposed. A piecewise linear model of time series sensor data is constructed by SSA. The model is a two-layered distributed detection model where the first layer is a local detection process at each node, whereas the second layer is a centralized detection process at the cluster head node. In [13], a distributed outlier detection method based on credibility feedback in WSNs is proposed. The method consists of three stages: evaluating the initial credibility of sensor nodes, evaluating the final credibility based on credibility feedback and Bayesian theorem, and adjusting for the outlier set. However, by the message complexity analysis, the energy consumption of this model is large.

2) CLUSTERING-BASED

In [14], a global approach based on a distributed non-parametric model is proposed. Each node clusters measurements using a fixed-width clustering algorithm and sends cluster results to its parent node. Then, the cluster head merges its children's cluster results and reports them to the sink node. Finally, the sink node detects potential outliers. If the average inter-cluster distance is greater than one standard deviation of the inter-cluster distance from the mean inter-cluster distance, it is identified as an anomalous cluster. In [15], a light-weight statistical detection method using K-means clustering is proposed. It also takes into account of sensor energy and processing power. In [16], a real-time outlier detection method in WSNs is introduced. The advantage of the method is that it requires no prior knowledge of the distribution of the data. However, the quality of the resulting clusters has a significant impact on the performance of outlier detection.

3) CLASSIFICATION-BASED

In [17], a threshold-free approach for outlier detection in industrial wireless sensor networks (IWSNs) is proposed. First, it utilizes the autoregressive model to obtain the predicted measurement as same as [11]. Then, it maps the residuals between actual measurements and predicted measurements into a high-dimensional feature space using one-class support vector machine (OCSVM). The OCSVM classifies the input residuals without requiring any threshold. In [18], another OCSVM-based method for outlier detection is proposed. It reduces the computational complexity in the training phase and the testing phase. However, if there are outliers in the training set, the performance of OCSVM is very poor. In [19], a two-layered outlier detection technique based on sensor fusion using the hierarchical structure of the network is proposed. The first layer uses the Bayesian classifier at a sensor node, then the decisions of individual sensor nodes are fused in the second layer to detect whether eventually an outlier is in the sensed data. It aims at providing an accurate outlier classification method that reducing computational complexity and communication overhead.

4) SPECTRAL DECOMPOSITION-BASED

In [20], a hierarchical anomaly detection model in distributed large-scale sensor networks is proposed. It exploits principal component analysis (PCA) to deal with outliers in data generated by the faulty sensor. The technique develops a model for the sensor data, and this model is used to detect outliers in a sensor node through neighbor sensor nodes. Since the technique requires the selection of the best model, it is computationally expensive. In [21], a distributed outlier detection model based on one-class principal component classifier (OCPCC) is proposed. It utilizes the spatial correlations among sensor nodes in a neighborhood. For each node in a cluster, a local normal reference model is constructed and sent to the cluster head. A cluster head receives local reference models and combines them to obtain a global normal reference model. The global normal reference model is sent back to each node for outlier detection. However, the false-positive rate for this model is high.

5) NEAREST NEIGHBOR-BASED

In [22], a local outlier factor (LOF) model is proposed. The average distance from a measurement to its nearest neighbors is denoted by d. The average of the nearest neighbors' distances to their nearest neighbors is denoted by $\overline{d}_n$. The LOF uses the ratio of $\frac{d}{\overline{d}_n}$ to detect outliers. In [23], a k-nearest neighbor (k-NN) based outlier detection scheme

for WSNs is proposed. The major intuition of this approach is hyper-grid. Through redefining anomaly from a hypersphere detection region (DR) to a hypercube DR, the computational complexity is reduced significantly. However, it needs to be validated on more datasets that possess dynamic changes in the data distribution.

6) OTHERS

With the development of ensemble learning [29], several methods [24], [25] based on the principle of isolation have been proposed. The intrinsic characteristics of anomalous data are fully considered to construct an outlier detector. It is known that an outlier is an instance which possesses attribute values different from normal measurements. In other words, outliers are ''few and different'', which makes them more isolated than normal measurements. In [26], a distributed outlier detection model based on isolation forest (iForest) for WSNs is proposed. This model does not use any distance or density metrics for outlier detection. Instead, local detectors of neighbor nodes are used to construct a global detector. The isolation score of each measurement is calculated by a global detector. However, since the iForest algorithm is insensitive to local outliers which are similar to normal measurements, the performance of this model is poor.

B. ISOLATION FOREST

In [25], a novel isolation method called Isolation Forest(iForest) is proposed. It assumes that outliers are few and different, and they are more susceptible to the isolation that the normal measurements. It means that outliers possess a sparse spatial distribution and are separated from a dense region. In a feature space, measurements in sparse regions are most likely to be outliers. Thus, it can be considered that the measurements in these regions are outliers. If a dataset is divided recursively and randomly, measurements in dense regions are hard to be isolated, which means many iterations are needed. On the contrary, measurements in sparse regions that need fewer iterations to be divided are easily isolated. To address this problem, iForest divides a dataset by a binary tree, namely an isolation tree (iTree). Measurements with shorter paths in an isolation tree are most likely to be outliers. The definition of iTree is as follows.

Definition 1 (Isolation Tree): Let T be a node of an isolation tree, and x be a measurement in a dataset. T is either a leaf node with no child, or a branch node with one test and exactly two child node T_l and T_r. A test consists of an attribute q and a split value p such that the test $x.q < p$ divides a measurement into T_l and T_r.

Given a data $X = \{x_1, x_2, \cdots, x_n\}$, there might be several attributes. For an attribute q, we randomly choose a split value p. The construction of an iTree is ended once one of the following three conditions are satisfied:

1) The height of the tree reaches a limit l;
2) There is only one measurement in X, namely $|X| = 1$;
3) The values of the measurements are the same.

In general, the performance of outlier detection of a single isolation tree is poor. Thus, the iForest consists of multiple isolation trees. The construction of the iForest is shown in Algorithm 1.

Algorithm 1 *iForest*(X, t, Ψ)

```
iForest ← ∅
l = ceiling(log2 Ψ)
for i = 1 to t do
    X_i = RandomSubset(X, Ψ)
    ConstructiTree_i
    iForest = iForest ∪ {iTree_i}
end for
return iForest
```

There are three parameters: the input measurements X, the number of iTrees t, and the number of measurements contained in each iTree Ψ. In addition, the upper bound of the height of the tree is calculated as $l = ceiling(\log_2 \Psi)$. The measurements in X are divided into t subsets and the number of measurements in each subset is Ψ. Then, the corresponding iTree of each subset is constructed. Finally, the algorithm returns the collection of iTrees, namely iForest.

In the detection phase of the iForest algorithm, the task is obtaining a ranking of the outlier scores of the measurements, namely the outlier scores are sorted in descending order. For a pre-defined threshold, a measurement whose outlier score is greater than the threshold is considered as an outlier. In other words, the measurements at the top of the ranking list are outliers. The outlier score of a measurement x is calculated as

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}}, \tag{1}$$

where $s(x, n)$ is the outlier score of measurement x, n is the number of measurements in X_i. $h(x)$ denotes the path length of x, which is the number of edges from the root node to x. For x, there are several values of $h(x)$ corresponding to the collection of iTrees in a iForest. $E(h(x))$ is the average of $h(x)$ corresponding to m iTrees, which is calculated as

$$E(h(x)) = \frac{1}{m}\sum_{i=1}^{m} h(x). \tag{2}$$

We employ the idea from Binary Search Tree(BST) to estimate the average path length of an iTree. For the n measurements in X_i, the average of path length of unsuccessful search in BST is

$$c(n) = 2H(n-1) - \frac{2(n-1)}{n}, \tag{3}$$

where $H(n)$ denotes the harmonic number which is estimated by $\ln(n) + 0.5772156649$, and the constant 0.5772156649 is the Euler's constant. We use $c(n)$ to normalize $h(x)$.

C. LOCAL OUTLIER FACTOR

In [22], a classic outlier detection strategy called Local Outlier Factor (LOF) is proposed. LOF is a density-based

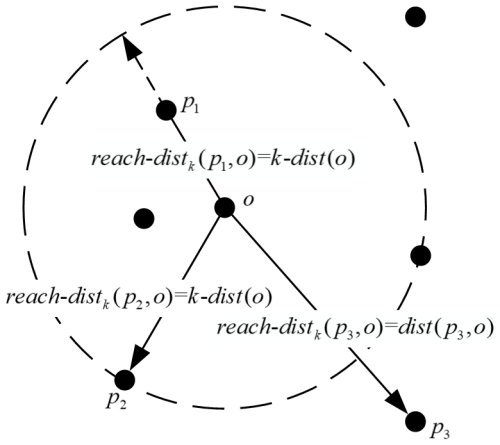


FIGURE 1. An example of the reachability distance.

method. It calculates the local density of a particular measurement and the local densities of the neighbors of the measurement. If these densities are similar, it is assumed that the particular measurement and its neighbors are in the same region, that is, they are in the same cluster. Specifically, if the average distance from a measurement to its nearest neighbors is denoted by d, the local density of the measurement is $\frac{1}{d}$. If the average of the nearest neighbors' distances to their nearest neighbors is denoted by $\overline{d}_n$, the average local density of the neighbors of the measurement is $\frac{1}{\overline{d}_n}$. The LOF is defined as the ratio of $\frac{d}{\overline{d}_n}$.

Definition 2 (k-Distance of a Measurement p): For any positive integer k, the k-distance of a measurement p is denoted as $k\text{-}dist(p)$. In specific, the value of $k\text{-}dist(p)$ is equal to $dist(p, o)$. $dist(p, o)$ is the distance between p and another measurement $o \in D$, such that: 1) $dist(p, o') \leqslant dist(p, o)$ holds for at least k measurements $o' \in D \setminus \{p\}$, and 2) $dist(p, o') < dist(p, o)$ holds for at most $k-1$ measurements $o' \in D \setminus \{p\}$.

By Definition 2, for a given $k\text{-}dist(p)$, the measurement o may be unique.

Definition 3 (k-Distance Neighborhood of a Measurement p): Given the k-distance of a measurement p, the k-distance neighborhood of a measurement p refers to the measurements whose distances between p are not greater than $k\text{-}dist(p)$, namely $N_k(p) = \{q \in D \setminus \{p\} \,|dist(p, q) \leqslant k\text{-}dist(p)\}$. These measurements q are called the k-nearest neighbors of p.

Definition 4 (Reachability Distance of a Measurement p With Respect to a Measurement o): Let k be a positive integer. The reachability distance of a measurement p with respect to a measurement o is defined as $reach\text{-}dist_k(p, o) = \max\{k\text{-}dist(o), dist(p, o)\}$.

As shown in Fig. 1, the dotted circle denotes the k-distance neighborhood of a measurement o. Measurements p_1 and p_2 are considered to be close to the measurement o. Thus, p_1 and p_2 are in the k-distance neighborhood of o. Namely, $reach\text{-}dist_k(p_1, o) = reach\text{-}dist_k(p_2, o) = k\text{-}dist(o)$. While the measurement p_3 is considered to be far away from o, thus the reachability distance of p_3 and o is simply the actual distance between them, namely $reach\text{-}dist_k(p_3, o) = dist(p_3, o)$. Therefore, the reachability distance $reach\text{-}dist_k(p, o)$ is at least the k-distance of o or the actual distance between p and o. In this way, the statistical fluctuation of $dist(p, o)$ for all the measurements p which are the neighbors of o can be significantly reduced. The strength of this smoothing effect can be controlled by the parameter k. The greater the value of k, the more similar the reachability distances for measurements within the same neighborhood.

Definition 5 (Local Reachability Density of a Measurement p): The local reachability density of a measurement p is defined as the reciprocal of the average reachability distance of measurement p in k-distance neighborhood, which is calculated as

$$lrd_k(p) = 1/\frac{\sum_{o \in N_k(p)} reach\text{-}dist_k(p, o)}{|N_k(p)|}. \tag{4}$$

The greater the density is, the more likely p and its neighborhood belong to a same cluster.

Definition 6 (Local Outlier Factor of a Measurement p): The local outlier factor of a measurement p is defined as

$$\mathrm{LOF}_k(p) = \frac{\sum_{o \in N_k(p)} \frac{lrd_k(o)}{lrd_k(p)}}{|N_k(p)|}. \tag{5}$$

The local outlier factor of a measurement p is the mean value of the ratio of the local reachability density of the measurement p analyzed to the local reachability density of its neighbors. The smaller the local reachability density of p and the greater the local reachability density of the neighborhood of p, the greater the value of $\mathrm{LOF}_k(p)$. In specific, when $\mathrm{LOF}_k(p) \in (0, 1+\sigma]$, p and its neighborhoods are belonged to the same cluster. For $\mathrm{LOF}_k(p) \in (1+\sigma, +\infty)$, p is considered as an outlier.

III. ISOLATION-BASED OUTLIER DETECTION FRAMEWORK USING NEAREST NEIGHBOR ENSEMBLES

A. PROBLEM STATEMENT

A wireless sensor network typically consists of a large number of sensor nodes scattered over a region of interest to monitor specific physical phenomena. Sensor nodes may be arranged to various kinds of topologies for different applications. A typical network topology is shown in Fig. 2: there are totally seven clusters in the network and clusters are separated by the dashed line. Each cluster has a cluster head node which is denoted by a grey point. Other nodes are denoted by black points. There is only one sink node, which is denoted by a black circle. A communication link between a cluster head node and the sink node is denoted by a dotted line.

To facilitate the description of our framework, we introduce the definition of a sub-network in a wireless sensor network.

Definition 7 (Sub-Network): A sub-network is one of the clusters in the whole network, in which nodes can directly

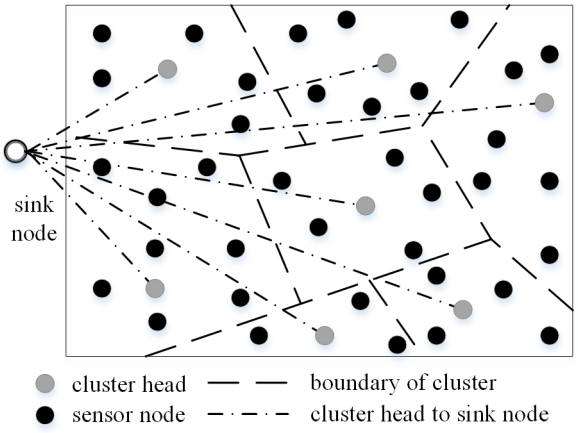


FIGURE 2. A typical topology of WSN.

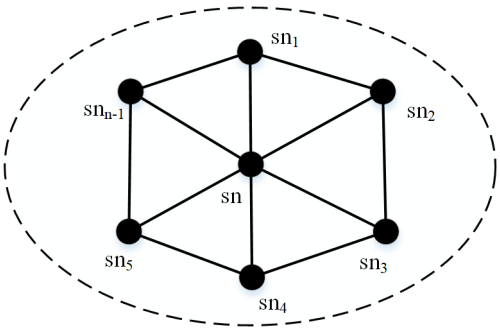


FIGURE 3. A sensor sub-network in WSN.

communicate with each other. Each node in the sub-network does the same work such as data collection, communication, and outlier detection. We denote a sub-network which contains n sensor nodes by $SN = \{sn, sn_1, \cdots, sn_{n-1}\}$. The sensor nodes densely deployed in a sub-network are homogeneous and time-synchronized. Thus, the sensor data in a sub-network tends to be correlated both in space and time.

A illustrative example of a sub-network is shown in Fig. 3. For a sensor node sn in a sub-network SN, the neighborhood of sn is denoted by set $N = SN \setminus sn$, where $|N|$ is $n-1$. Each sensor node in the sub-network produces an observation vector consists of multiple attributes, which is denoted by $X = (x_1, x_2, \cdots, x_d)$, where d is the number of attributes.

In this paper, we propose a new framework based on the above sub-network to detect outliers for WSNs. In the next subsection, we will first introduce our outlier detection algorithm.

B. ISOLATION USING NEAREST NEIGHBOR ENSEMBLES

The main idea of the isolation using nearest neighbor ensemble (iNNE) [31] is as follows. For an instance x in the training set, the isolation of x is implemented by building a hypersphere that only covers x. The radius of the hypersphere is determined by the distance between x and its nearest neighbor (NN) in the training set. Therefore, if x locates in a sparse area, the corresponding hypersphere is large. On the contrary, if x locates in a dense area, the corresponding hypersphere is small. In general, outlier instances appear in sparse areas, while normal instances appear in dense areas. Thus, the radius of the hypersphere can be used to detect outliers. The details of the iNNE are as follows.

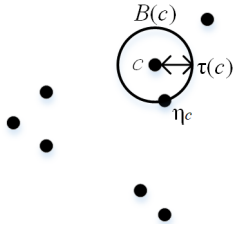


FIGURE 4. A random selected subset S of size $\Psi = 8$.

For a given dataset $D \subset \mathscr{R}^d$, let $\|a-b\|$ denotes the Euclidean distance between a and b, where $a, b \in \mathscr{R}^d$. For instance x, the nearest neighbor(s) of x are denoted by a set $N_x = \eta_{x1}, \eta_{x2}, \cdots, \eta_{xn}$, where $n \geqslant 1$. For a subset $S \subset D$, we denote $\Psi = |S|$. Similar as iForest algorithm, Ψ also denotes the number of instances in a subset.

Definition 8: A hypersphere $B(c)$ centered at c with the radius of $\tau(c) = \|c - \eta_c\|$.

The hypersphere $B(c)$ isolates c from the other instances in S. Its radius $\tau(c)$ is a measurement of the degree of isolation of c. The larger the radius is, the more isolated c is, and vice versa. We choose to use a local measurement to show the degree of isolation, which considers the relative isolation score of $B(c)$ and $B(\eta_c)$.

Definition 9: The isolation score for $x \in \mathscr{R}^d$ for set S is

$$I(x) = \begin{cases} 1 - \dfrac{\tau(\eta_{cnn(x)})}{\tau(cnn(x))}, & \text{if } x \in \bigcup\limits_{c \in S} B(c) \\ 1, & \text{otherwise}, \end{cases} \tag{6}$$

where $cnn(x) = \arg\min\{\tau(c) : x \in B(c), c \in S\}$. As $\frac{\tau(\eta_{cnn(x)})}{\tau(cnn(x))} \leqslant 1$, $I(x)$ is in the range of $(-\infty, 1]$. When none of the hyperspheres covers x, then x is very far from all points in S. Thus, $I(x)$ is 1. If $I(x)$ is very close to 1, x could be considered as an outlier. Here, we give an illustrative example to illustrate the iNNE algorithm.

As shown in Fig. 4, there is a random selected subset S of 8 instances extracted from a dataset. Each instance of this subset is used as the center of the hyperspheres created. In addition, Fig. 4 also shows an example of hypersphere $B(c)$ created using c with radius $\tau(c)$.

As shown in Fig. 5, there are all the 8 hyperspheres created for the subset S of 8 instances. This set of hyperspheres is used for the calculating isolation scores for the 2 instances y, $z \in \mathscr{R}^d$. To compute the isolation score for z, 2 hyperspheres need to be determined: the smallest hypersphere (has a center at a) that covers z and the hypersphere centered at the NN of a in the subset S. The isolation score $I(z)$ is determined based on the ratio of the radius of the two hyperspheres, namely, $\tau(a)$ and $\tau(\eta_a)$. In contrast, instance y does not fall in any

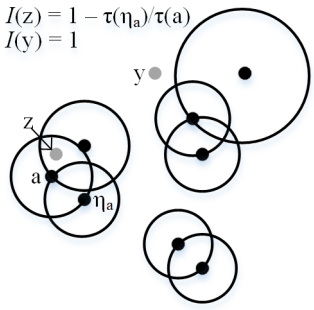


FIGURE 5. Isolation scores determined for *y*,*z*.

hyperspheres. Thus, it obtains the maximum isolation score, namely 1.

Definition 10: The iNNE has t sets of hyperspheres, which are generated from t subsets S_i, where $i = 1, 2, \cdots, t$. The t sets of hyperspheres are

$$\{\{B(c) : c \in S_i\} : i = 1, 2, \cdots, t\}. \tag{7}$$

Definition 11: The outlier score for $x \in \mathscr{R}^d$ is defined as

$$\bar{I}(x) = \frac{1}{t}\sum_{i=1}^{t} I_i(x), \tag{8}$$

where $I_i(x)$ is the isolation score based on subset S_i.

The isolation using nearest neighbor ensembles contains two stages: training stage and evaluation stage. In the training stage, t sets of hyperspheres are built from t subsets. The number of elements of each subset is Ψ. Details of the training stage can be found in Algorithm 2. There are three input parameters In Algorithm 2. D is a training dataset, Ψ is the size of a subset, and t is the number of subsets. Similar to the iForest algorithm, the instances in D are divided into t subsets. Then, the corresponding hyperspheres are constructed for each instance in a subset by definition 8. Finally, the algorithm returns the set of these hyperspheres, namely iNNE. As we need to compute the distance between two instances, the time complexity of the training stage is $O(t\Psi^2)$. The memory complexity in the training stage is dominated by t sets of hyperspheres, so the memory requirement is $O(t\Psi)$, where t and Ψ are constants. Thus, iNNE in the training stage has constant complexity.

Algorithm 2 *iNNE_generation*(D, t, Ψ)

```
iNNE ← ∅
for i = 1 to t do
    S_i = RandomSubset(D, Ψ)
    B_i ← ∅
    for i = 1 to ‖S_i‖ do
        B_i = B_i ∪ {B(c)}
    end for
    iNNE = iNNE ∪ {B_i}
end for
return iNNE
```

In the evaluation stage, each test instance is evaluated against t sets of hyperspheres in the iNNE. The isolation scores are obtained by Definition 9, and the outlier scores are obtained by Definition 11. The detailed process of the evaluation stage can be seen in Algorithm 3. There are two input parameters: evaluation model iNNE and evaluation dataset D_E. The algorithm outputs the outlier scores of instances in D_E. For a test set that contains n test instances, a distance between each test instance and each instance in the t training sets of hyperspheres needs to be calculated. Thus, the time complexity and memory complexity of the evaluation stage are both $O(nt\Psi)$, which is linear with respect to n. Thus, iNNE in the evaluation stage has linear complexity.

Algorithm 3 *iNNE_evaluation*$(iNNE, D_E)$

```
Score(D_E) ← ∅
for i = 1 to |D_E| do
    for j = 1 to iNNE.t do
        Calculate I_j(x)
    end for
    Calculate Ī(x)
    Score(D_E) ← Score(D_E) ∪ Ī(x)
end for
return Score(D_E)
```

C. THE DISTRIBUTED DETECTION FRAMEWORK BASED ON INNE FOR WSNS

In this subsection, we will introduce our distributed detection framework based on iNNE for WSNs. In order to reduce the communication burden in WSNs, each sensor node possesses a local outlier detector based on training datasets. Each sensor node broadcasts the information given by its local detector to its neighbor sensor nodes. This information includes hypersphere structures and current measurements. The operation of the global detector of a sensor node is based on the information of its local detector and the information from all its neighbor sensor nodes. As shown in Fig. 6, *sn* is a sensor node of the sub-network and its neighbor sensor nodes are sn_1, sn_2, sn_3, and sn_4. The detection of outliers in *sn* is conducted by the global detector.

Specifically, Fig. 7 depicts our proposed framework, which is divided into three phases: *training phase*, *detection phase*, and *update phase*.

1) TRAINING PHASE

Before the on-line detection, off-line training is performed to obtain an initial outlier detector. In this phase, a local detector is constructed using the historical measurements of each sensor node based on the iNNE. The local detectors constructed during the training phase are used to construct global detectors by various combinations, such as weighted voting [32], bagging [33], and stacked generalization [34]. Among them, the weighted voting scheme is suitable for large-scale integration. It is usually used when the

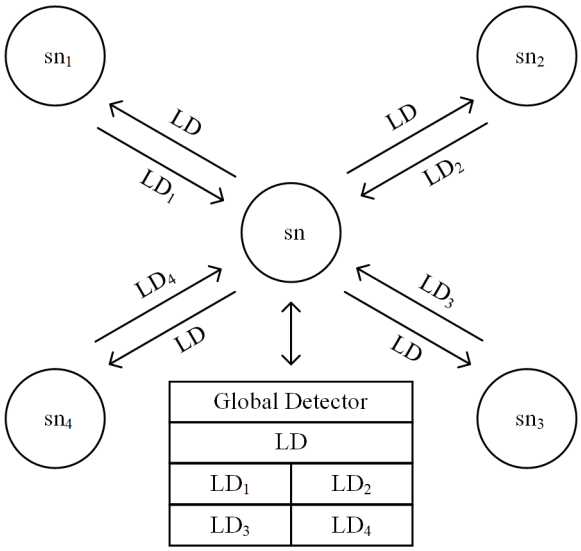


FIGURE 6. An abstract view of the proposed model.

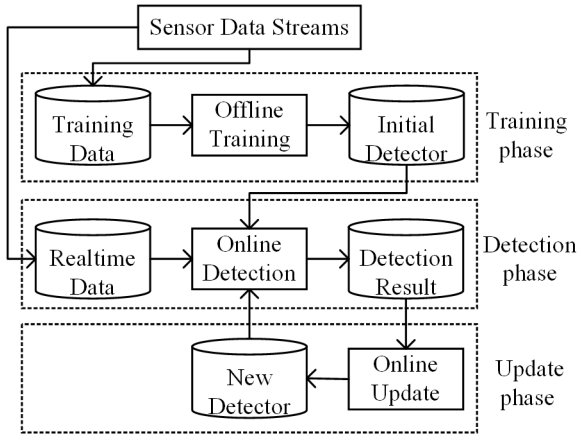


FIGURE 7. Three phases of the proposed framework.

performance of an individual local detector varies significantly. For bagging and stacked generalization, a lot of computations are needed, which is a heavy load for wireless sensor networks. Thus, we propose a new combination method based on weighted voting to combine multiple local detectors. The principle of voting is that multiple detectors process the same input in parallel, then the outputs are combined based on the spatial correlations among sensor nodes. The actual distances between sensor nodes are used as the weight. The closer the distance is, the greater the weight of the neighbor sensor is, and vice versa. The details of the weight setting are described in the detection phase. Since the voting method just requires the transmission of the current measurements of neighbor nodes, the broadcast of the hypersphere structures of local detectors is avoided. This feature significantly reduces the amount of data transmission. The procedure of the training phase is the same as the Algorithm 2, where the dataset D refers to the historical measurements of each sensor node.

2) DETECTION PHASE

The detection phase is carried out online at each node. The initial detector of the training phase is used at the beginning of the detection phase. As time goes on, the update of the detector effectively maintains the accuracy of our framework. To effectively detect outliers in our framework, we propose a new combination method based on weighted voting to combine local detectors. In specific, when a new measurement x is obtained by sensor node sn, the local detector of sn calculates its outlier score. Meantime, the new measurement x is broadcasted to the neighbor sensor nodes of sn. The local detectors of the neighbors calculate the outlier scores. Then the neighbors send the outlier scores and the positions of the neighbors back to sn. Finally, the global outlier score of x is calculated as

$$I_g(x) = \frac{wI(x) + \sum_{i=1}^{n} w_i I_i(x)}{w + \sum_{i=1}^{n} w_i}. \tag{9}$$

In (9), n is the number of neighbor sensor nodes. $I_i(x)$ is the outlier score calculated by the local detector of the neighbor sensor node sn_i. $I(x)$ is the outlier score calculated by the local detector of sensor node sn. w_i is the weight of the neighbor sensor node sn_i, it is denoted based on the Euclidean distance between sn and sn_i which is the reciprocal of $\|pos - pos_i\|$. Namely, $w_i = \frac{1}{\|pos - pos_i\|}$. w is the weight of sensor node sn, it is calculated as $w = \sum_{i=1}^{n} w_i$. For $w, w_1, w_2, \cdots, w_i, \cdots, w_n$, we set the weight of sn to the sum of the weights of all neighbor sensor nodes. By Definition 9, $I(x)$ is in the range of $(-\infty, 1]$. Thus, the global outlier score $I_g(x)$ which is similar to $I(x)$ in (6) is also in the range of $(-\infty, 1]$. Then, a measurement x is determined by the following principles, where σ is a small positive number.

1) If $I_g(x)$ is close to 1, which also means that if $I_g(x) \in [1 - \sigma, 1]$, then x is an outlier;

2) If $I_g(x) \leqslant 0$, then x is a normal measurement;

3) If $I_g(x) \in (0, 1 - \sigma)$, then x is either an outlier or a normal measurement.

However, in most cases, the obtained global outlier score is in the range of $(0, 1 - \sigma)$. Therefore, it is not necessary to choose the value of σ. We directly introduce a threshold p to label a measurement x:

$$label(x) = \begin{cases} 0, & \text{if } I_g(x) < p \\ 1, & \text{if } I_g(x) \geqslant p, \end{cases} \tag{10}$$

where 0 indicates a normal measurement, while 1 refers to an outlier. In practice, the threshold p has a great effect on the results of outlier detection. However, the value of p is always unknown and this is a non-trivial problem. In fact, a roughly estimated range is given based on prior knowledge. In this paper, the value of p is investigated by experiments.

The detailed detection phase is shown in Algorithm 4. There are seven parameters. For sensor node sn, x is a new measurement, LD is the local detector and LD_i is the local detector of a neighbor sensor node sn_i. pos refers to the real

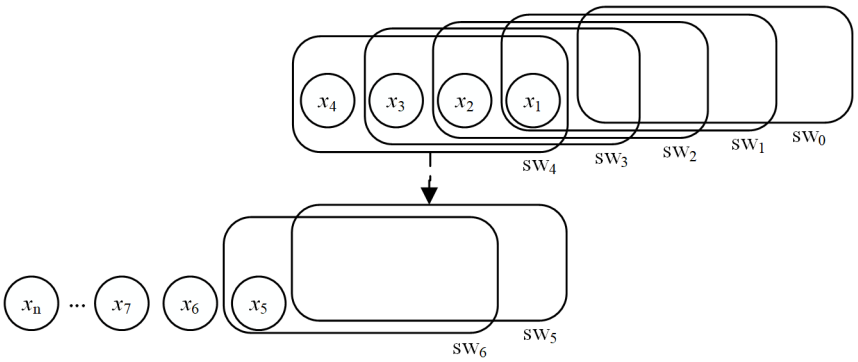


FIGURE 8. The sliding window.

position of sn, while pos_i refers to the real position of sn_i. n is the number of neighbor sensor nodes of sn. p is the threshold.

Algorithm 4 *Detection*$(x, LD, LD_i, pos, pos_i, p, n)$

$I(x) \leftarrow iNNE\_evaluation(LD, x)$
for $i = 1$ **to** n **do**
  $I_i(x) \leftarrow iNNE\_evaluation(LD_i, x)$
  $w_i = \frac{1}{\|pos - pos_i\|}$
end for
return $(I_g(x) \geqslant p)?1 : 0$

3) UPDATE PHASE

The update phase retrains the outlier detection model for the purpose of adapting the dynamic changes of sensor data. In specific, we employ a sliding window to process sensor data. The size of the window is fixed. Briefly, every time the sliding window is filled with a new group of data, the outlier detection model is retrained. Thus, the model is highly consistent with the recent data distribution. This feature effectively improves the accuracy of detection.

As shown in Fig. 8, the data stream is represented by a series of circles $x_1, x_2, \cdots, x_n$. For simplicity, the size of the sliding window is four. There are totally six states of the sliding window. The initial state of the slide window is sw_0, which is an empty window. At this time, the corresponding detection model is DM_0, which is able to calculate a global outlier score $I_g(x)$. The sw_1 contains one measurement x_1. For sw_1, the corresponding detection model is also DM_0. Thus, x_1 is detected by DM_0. If x_1 is an outlier, it is removed from sw_1. On the contrary, if x_1 is a normal measurement, it is retained in sw_1. Actually, for sw_i, $i = 1, 2, 3, 4$, the corresponding detection model is DM_0. For sw_4, if x_4 is not an outlier, the sliding window is full. Then, the detection model is retrained by Algorithm 2 using x_1, x_2, x_3, and x_4. The new detection model is denoted by DM_1. Then, the sliding window is emptied. Namely, the corresponding detection model of sw_5 is DM_1. Every time the sliding window is full, the detection model is retrained.

The detailed update phase is shown in Algorithm 5. There are four parameters. x is a vector that contains n measurements and m is the size of the sliding window. LD is the local detector. And $buffer$ stores normal measurements which are used to retrain the local detector.

Algorithm 5 *Update*$(x, m, LD, buffer)$

for $i = 1$ **to** n **do**
  if $buffer.length < m$ **then**
    if $I_g(x_i) < p$ **then**
      $buffer \leftarrow buffer \cup x_i$
    end if
    break
  else
    $LD \leftarrow iNNE\_generation(buffer, LD.t, LD.\Psi)$
    $buffer \leftarrow \varnothing$
  end if
  return LD
end for

IV. EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we evaluate the effectiveness of our framework. Experiments are conducted on a personal PC with Intel Core i5-4200M 2.50 GHz, which runs Microsoft Windows 10 Professional (64-bit) with 8GB RAM. Meanwhile, the following software utilities are used for the preparation of data samples, implementation of algorithms and the analysis of the results: Matlab R2016a.

A. DATASET

Two datasets are used to evaluate the proposed framework: ISSNIP [35] and IBRL [36].

1) ISSNIP

The Intelligent Sensors, Sensor Networks & Information Processing(ISSNIP) dataset is a real humidity-temperature sensor data that is collected using TelosB motes in a single-hop WSNs. This dataset has controlled outliers, and all the data in the dataset are labeled. There are totally

TABLE 1. Summary of the ISSNIP dataset.

Node	Setting	Total samples	Normal	Outlier
1	Indoor	4417	4300	117
2	Indoor	4417	4417	0
3	Outdoor	5039	5039	0
4	Outdoor	5041	5009	32

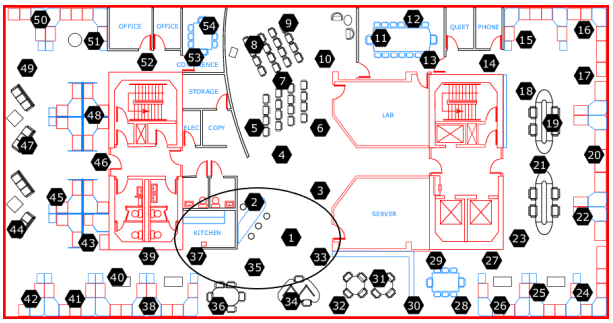


FIGURE 9. The deployment of sensor nodes in IBRL.

four sensor nodes: two indoor sensor nodes and two outdoor sensor nodes. The data consists of temperature and humidity measurements collected over a period of 6h at the interval of 5s. In order to generate outlierss, a hot-water kettle is used to increase the temperature and the humidity simultaneously. Thus, all outliers in ISSNIP are event outliers since they are a sequence of errors or erroneous readings caused by an event in the dataset. The summary of the dataset is shown in Table 1.

2) IBRL

The Intel Berkeley Research Lab (IBRL) dataset is based on the publicly available Intel Lab Data consisting of real measurements collected from 54 sensors deployed at the Intel Berkeley Research Lab. Mica2Dot sensors with weatherboards collected timestamped topology information, along with humidity, temperature, light and voltage values once every 31 seconds. The deployment of sensor nodes in IBRL is shown in Fig. 9 [36].

We consider five sensor nodes 1, 2, 33, 35, and 37 in the WSN and select the temperature and humidity measurements recorded from Feb. 28 to Feb. 29, 2004. As these data are not annotated and contain no labeled information, a data preprocessing is needed before the evaluation. Firstly, the data is manually cleaned by removing spurious measurements (obviously, extreme values are outliers) with the help of a scatter diagram. The cleaned data are labeled as normal. Secondly, artificial outliers which are normally distributed are randomly injected to the normal data for each node. The amount of injected outliers are 3% of the normal measurements. In addition, they are manually labeled as outliers. Without loss of generality, we make the differences in the mean and standard deviation between the artificial outliers and the normal data are slight. For example, in sensor node 1 of IBRL dataset, the mean temperature of normal data is 21.03 and the standard deviation of them is 2.44. Then we choose the mean temperature of artificial outliers to $21.03 + 1$ and choose the standard deviation of them to $2.44 + 0.5$. Finally, the artificial outliers are generated using the normal random distribution function based on these two values, namely, Artificial outliers $= f(22.03 + 1, 2.44 + 0.5, n)$, where f is the random number generator based on the normal measurements' distribution, n is the number of desired artificial outliers. As the outliers in IBRL are generated randomly, most of them are discrete outliers. The summary of the preprocessed IBRL dataset is shown in Table 2.

TABLE 2. Summary of the preprocessed IBRL dataset.

Node	Pos(x)	Pos(y)	Total samples	Normal	Outlier
1	21.5	23	2300	2231	69
2	24.5	20	2300	2224	76
33	19.5	26	2300	2100	100
35	24.5	27	2300	2239	61
37	27.5	26	2300	2261	39

B. PERFORMANCE METRICS

In order to evaluate our proposed framework, we select three performance metrics: accuracy rate (ACC), detection rate (DR), and false alarm rate (FAR). They are calculated as

$$\mathrm{ACC} = \frac{\mathrm{TN} + \mathrm{TP}}{\mathrm{TN} + \mathrm{FP} + \mathrm{TP} + \mathrm{FN}} \times 100\% \tag{11}$$

$$\mathrm{DR} = \frac{\mathrm{TP}}{\mathrm{FN} + \mathrm{TP}} \times 100\% \tag{12}$$

$$\mathrm{FAR} = \frac{\mathrm{FP}}{\mathrm{TN} + \mathrm{FP}} \times 100\%, \tag{13}$$

where TP, FP, TN, and FN denote the number of correctly detected outliers, the number of normal measurements which are wrongly detected as outliers, the number of correctly detected normal measurements, and the number of outliers wrongly detected as normal, respectively.

Another metric is Area Under Curve (AUC). Compared with ACC, DR, and FAR, AUC objectively reflects the ability of a detector regardless of the value of threshold. The AUC is calculated as

$$\mathrm{AUC} = \frac{\sum_{i=1}^{M} r_i - \frac{M \times (M+1)}{2}}{M \times N}, \tag{14}$$

where M and N denote the numbers of outliers and normal measurements labeled before the experiments, respectively. The outlier scores of the $M + N$ measurements are sorted in ascending order. The r_i is the index of the i-th pre-labeled outlier.

C. EXPERIMENTS AND ANALYSIS

1) EVALUATION OF iNNE

In order to evaluate the performance of the iNNE, we compare it with iForest [25] and LOF [22] in terms of detection accuracy and sensitivity to the parameter selection. In specific, 70% data of the ISSNIP dataset are used for training. The other 30% data are used for evaluating.

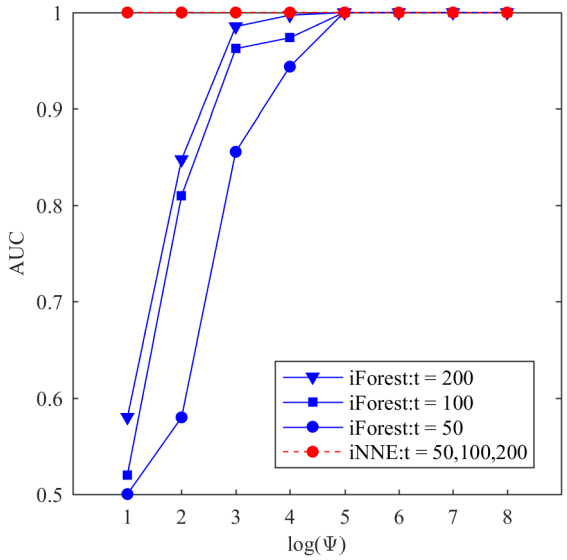


FIGURE 10. AUC for the iNNE and the iForest.

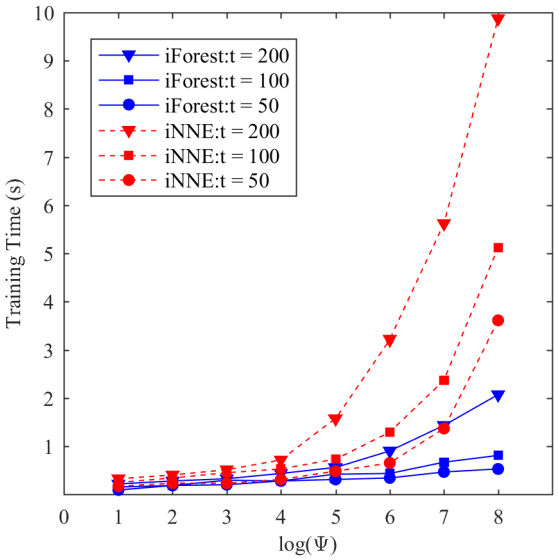


FIGURE 11. Training time for the iNNE and the iForest.

Since the iNNE inherits the concept of isolation from iForest, the two algorithms have the same parameters:the number of subsets t and the size of subsets Ψ. In specific, the number of subsets $t = 50$, 100, and 200; the size of subsets $\Psi = 2$, 4, 8, 16, 32, 64, and 128. The AUC and training time for the two algorithms are depicted in Fig. 10 and Fig. 11.

As shown in Fig. 10, the subset size Ψ is the number of measurements in a subset. For the iForest algorithm, the value of $\log\Psi$ is the height of the iForest. When the subset size $\Psi \in [2, 32]$, the value of AUC increases with an increase of Ψ. For $\Psi \geqslant 32$, the value of AUC keeps at 1. The higher the iTree, the more detailed the measurements in the dataset are divided. Thus, the outlier detection performance of iForest improves with the increase of Ψ. In addition, we set the number of iTrees in the iForest as for $t = 50, 100, 200$. In general, the outlier detection performance of the iForest improves with the increase of t. However, some iTrees in the iForest are likely to be similar. Thus, the improvement is not significant. For the iNNE algorithm, the values of AUC for $t = 50, 100, 200$ keep at 1 when the subset size $\Psi \in [2, 256]$. Thus, the iNNE outperforms the iForest. Besides, the performance of the iNNE is insensitive to the number of subsets t and the subset size Ψ. On one hand, the calculation of the outlier score in the iNNE algorithm uses the distances between the measurements, while the iForest algorithm uses the path lengths in the iForest. Thus, the iNNE does not require a lot of measurements to construct the tree structure in the iForest. On the other hand, when the number of measurements is small, distance is more important than path length in terms of reflecting the degree of the anomaly for a measurement. Thus, the iNNE outperforms the iForest when Ψ is small, namely $\Psi \in [2, 32]$. With the increase of the subset size Ψ, the distance between a measurement and its nearest neighbor in a larger subset is more reflective than a small subset in terms of the degree of the anomaly. Thus, for the iNNE algorithm, the value of AUC keeps at 1 for $\Psi \in [2, 256]$.

As shown in Fig. 11, with an increase of Ψ, the training times of both the iNNE and the iForest are increasing. As mentioned above, the time complexities of the iNNE and the iForest are $O(t\Psi^2)$ and $O(t\Psi)$, respectively. When $2 \leqslant \Psi < 32$, the difference of the training times between the iNNE and the iForest can be considered as slight. For $\Psi \geqslant 32$, the training time of the iNNE increases dramatically than that of the iForest. Similarly, With the increase of t, the training times of both the iNNE and the iForest are also increasing. Furthermore, for a small value of Ψ (e.g. $2 \leqslant \Psi \leqslant 16$), the iNNE possesses a good outlier detection performance with short training time. In addition, the memory requirement of the iNNE for a small Ψ is also small. Thus, it can be considered that the iNNE is a light-weight outlier detection algorithm.

To conduct a further investigation of our framework, we compare the iNNE with the Local Outlier Factor (LOF). The reason why we choose LOF is that both the iNNE and the LOF are based on the idea of the nearest neighbor. The parameter k in the LOF refers to a positive integer which is used to define the k-distance and the k-distance neighborhood of a measurement. The greater the value of k, the more measurements are in the k-distance neighborhood. Therefore, the role of the parameter k in the LOF and the parameter Ψ in the iNNE are similar. For the iNNE, we set $t = 50$. The AUC and the training time for the two algorithms are depicted in Fig. 12 and Fig. 13, respectively.

As shown in Fig. 12, with an increase of k, the AUC of the LOF increases until it reaches 1. This is because the greater the value of k is, the more measurements are in the k-distance neighborhood. In this case, the local densities of the measurements in the LOF are more accurate. Thus, the AUC of LOF improves. For the iNNE, as it uses the subset ensembles

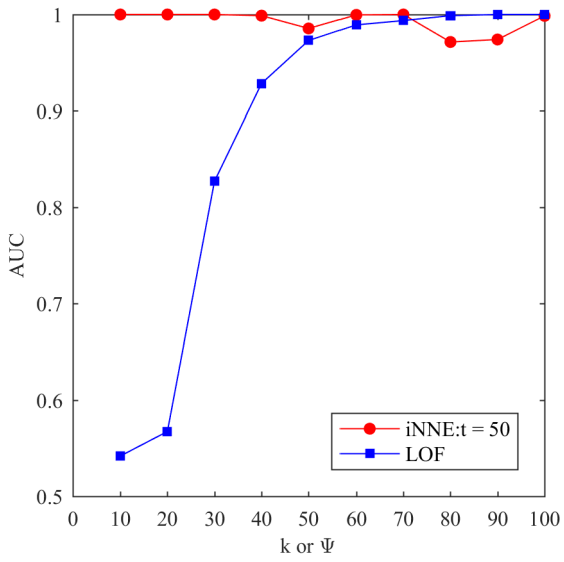


FIGURE 12. AUC for the iNNE and the LOF.

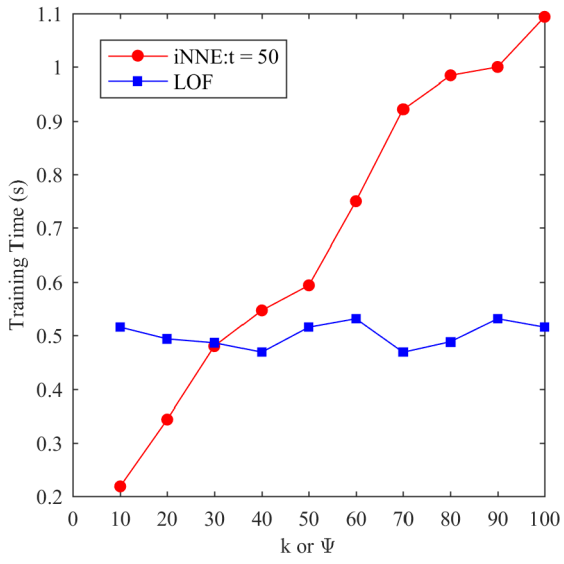


FIGURE 13. Training time for the iNNE and the LOF.

which improve the performance of outlier detection, the value of the AUC is greater than 0.95 for $\Psi \in [10, 100]$. Therefore, the performance of the iNNE is better than the LOF in most cases.

As shown in Fig. 13, with an increase of Ψ, the training time of the iNNE increases dramatically. On the contrary, the training time of the LOF maintains a relatively steady trend with some fluctuations. This is because the time complexity of the iNNE is $O(t\Psi^2)$, while the time complexity of the LOF is mere $O(n^2)$.

Here, n in the LOF is the number of measurements in the dataset. Thus, the time complexity is irrelevant to the parameter k. In addition, when Ψ is smaller than 30, the training time of the iNNE is shorter than that of the LOF. This is because $O(t\Psi^2)$ is smaller than $O(n^2)$ when Ψ is small. With the increase of Ψ, $O(t\Psi^2)$ becomes greater

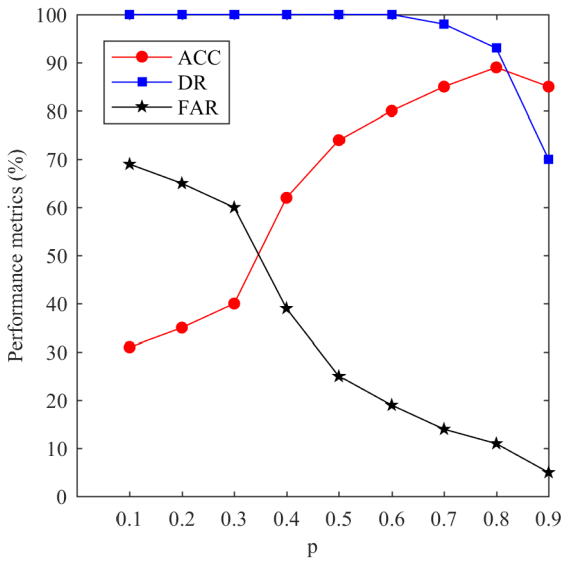


FIGURE 14. ACC, DR, and FAR for different thresholds.

than $O(n^2)$. Thus, when Ψ is greater than 30, the training time of the iNNE is greater than that of the LOF. In a nutshell, the iNNE outperforms the LOF when Ψ is small; while a large Ψ indicates a better performance of the iNNE instead of the LOF.

2) EVALUATION OF OUR PROPOSED FRAMEWORK

In order to evaluate the effectiveness of our proposed framework, we conducted extensive experiments on the ISSNIP dataset and the IBRL dataset. In addition, an improved isolation-based distributed outlier detection framework [26] is used to compare with the proposed framework.

Evaluation on ISSNIP Dataset: As mentioned above, the threshold p has a significant impact on the detection framework. If p is not appropriate, it will cause a large deviation and reduce the reliability of our framework. Thus, we first discuss the value of the threshold of p by experiments. We select 500 measurements from the ISSNIP dataset to evaluate the effectiveness of our framework with the different thresholds of p. The experimental results are shown in Fig. 14.

As shown in Fig. 14, with the increase of threshold p, the value of FAR is decreasing. When the threshold $p \in [0.1, 0.8]$, the value of DR decreases slightly while the value of ACC increases significantly with an increase of p. When the value of threshold p is set to 0.9, the values of ACC and DR decrease. Thus, when the value of the threshold p is small, though the value of ACC is 100%, the value of FAR is high and the ACC is low. It means that there are many normal measurements are detected as outliers. On the contrary, when we set a large value of threshold p, the values of DR and ACC are low. This is because that many outliers are not be detected. An appropriate threshold of p should enable our framework to achieve high ACC and DR with low FAR. In experiments, when p is set to 0.8, the value of ACC is the best. In addition, the DR and FAR also have good results.

TABLE 3. Experimental results on ISSNIP dataset (indoor).

framework	Window Size	Ψ	ACC(%)	DR(%)	FAR(%)
iNNE	100	8	**93.8**	48.7	5.4
iNNE	100	16	93.7	48.7	5.0
iNNE	100	32	**93.8**	48.7	**4.9**
[26]	100	64	67.8	46.4	11.8
iNNE	200	8	90.0	**100**	10.4
iNNE	200	16	90.6	**100**	9.7
iNNE	200	32	91.4	**100**	8.9
[26]	200	64	85.6	68.4	13.9
iNNE	300	8	87.2	48.7	11.6
iNNE	300	16	89.6	48.7	9.2
iNNE	300	32	89.8	48.7	8.9
[26]	300	128	74.6	23.2	56.4
[26]	300	256	82.8	0	14.8

TABLE 4. Experimental results on ISSNIP dataset (outdoor).

framework	Window Size	Ψ	ACC(%)	DR(%)	FAR(%)
iNNE	100	8	98.2	**96.8**	1.8
iNNE	100	16	98.3	**96.8**	1.5
iNNE	100	32	**98.7**	**96.8**	**1.3**
[26]	100	64	96.2	59.4	3.6
iNNE	200	8	96.4	**96.8**	3.6
iNNE	200	16	97.3	**96.8**	2.7
iNNE	200	32	**98.7**	**96.8**	3.6
[26]	200	64	92.4	65.7	3.6
iNNE	300	8	93.6	93.8	6.4
iNNE	300	16	94.5	93.8	5.5
iNNE	300	32	94.9	93.8	6.3
[26]	300	128	98.0	53.1	1.8
[26]	300	256	94.8	62.5	4.9

Therefore, the threshold p of our framework is set to 0.8 in the following experiments.

Due to that ISSNIP dataset is divided into indoor and outdoor, we consider node 1 and node 2 as a sub-network, and node 3 and 4 as another one. Since there are outliers only in node 1 and node 4. So we evaluate the effectiveness of our framework and [26] in node 1 and node 4 with different parameter combinations. For the number of subsets t, its value is set as 100 for the two frameworks. Similarly, for window size m, its value is set as 100,200 and 300. Subset size Ψ is set to 64,128,256 and threshold p is set to 0.7 in [26]. While in our framework, Ψ is set to 8,16,32 and p is set to 0.8. The experimental results for node 1 and node 4 are shown in Table 3 and Table 4. The bold-faced values are the best-reported results in all the window sizes.

As shown in Table 3, it can be seen that when the window size is set to 200, the proposed framework and [26] have their best results in DR. The DR of the proposed framework is 100%, whereas [26] is 68.4%. In this case, our framework also outperforms [26] in ACC and FAR. When the window size is 100 or 300, our framework has good values for ACC and FAR, while DR is very low. While the performance of [26] is poor. The ACC is lower than 80% and the DR is lower than 50%. Thus, our framework is better than [26].

As shown in Table 4, our framework and [26] both perform well in ACC, which are higher than 90% in all the window sizes. Moreover, our framework has good results for all the performance metrics with different combinations of parameters. Especially, when the window size is 100 and the subset is 32, all the performance metrics are the best in Table 3. While for [26], though the ACC is high and FAR is low, the DR keeps around 60%, which is much lower than the DR of iNNE. Thus, our framework is better than [26].

Furthermore, Table 3 and Table 4 show that the results of the proposed framework are mainly affected by the window size and insensitive to the subset size. For example, when the window size is 200 in Table 3, the DR of our framework is 100% all the time with the change of subset size. Its ACC and FAR have slight changes in this case. In addition, [26] behaves badly in these two nodes. This is because the outliers are generated by an event in the ISSNIP. Since [26] based on iForest is insensitive to the outliers generated by the event, it has a poor performance. On the contrary, our framework based on iNEE has good results. It proves that our framework outperforms [26] in detecting event outliers.

Evaluation on IBRL Dataset: We repeated the experiments on the IBRL dataset from 5 nodes in the WSNs. And [26] is still used to compare with our framework for different window sizes. The value of t is set to 100, subset size Ψ and threshold p are both set to the same values of the experiments on the ISSNIP dataset. The results of [26] and our framework on IBRL dataset are presented in Table 5. The bold-faced values are the best-reported results for each node in the table.

As shown in Table 5, the results indicate that all the performance metrics of both our framework and [26] get worse with the increase of window size. For node 1, as window size increases, the ACC of our framework decreases from 97.7% to 78.9%, DR decreases from 96.4% to 78.6%, and FAR increases from 2.3% to 21.4%. Similarly, the ACC and DR of [26] decrease significantly while FAR increases. Thus, the two frameworks both perform best when the value of window size is 100. In addition, the results in Table 5 also show that our framework outperforms [26] for all nodes in terms of detection accuracy. For node 2, when window size is 100, the ACC of [26] is 91.7%, DR is 97.0%, and FAR is 8.4%. Whereas the ACC of our framework is 99.9%, DR is 97.7%, and FAR is 0%, which is a satisfactory result for outlier detection.

Furthermore, all the performance metrics of our framework and [26] on the IBRL dataset perform much better than that metrics on the ISSNIP dataset. This is due to outliers in the IBRL dataset are randomly generated at a different time and most of them are discrete outliers. Since our framework and [26] both belong to isolation-based frameworks and are sensitive to these discrete outliers, they perform well on the IBRL dataset. In conclusion, our framework is able to detect discrete outliers and event outliers. And it performs better than the existing isolation-based outlier detection frameworks.

To show the viability in performance effectiveness of the combination methods in our framework, we compared the detection performances between a local detector and a uniform weight strategy. For IBRL dataset, we set the value of window size is 100, the number of subsets t is set to 100, subset size Ψ is set to 8, and threshold p is set to 0.7. The performances of different combination methods are presented in Table 6, Table 7, and Table 8.

TABLE 5. Experimental results on IBRL dataset.

Node	Window Size	iNNE ACC(%)	iNNE DR(%)	iNNE FAR(%)	[26] ACC(%)	[26] DR(%)	[26] FAR(%)
1	100	**97.7**	**96.4**	**2.3**	95.3	**96.4**	4.8
	200	89.8	78.6	9.9	88.5	80.4	11.3
	300	78.9	78.6	21.4	66.7	82.1	33.7
2	100	**99.9**	**97.7**	**0**	91.7	97.0	8.4
	200	90.8	86.4	9.1	91.1	81.8	8.6
	300	93.2	83.3	6.4	96.3	74.2	2.9
33	100	**94.1**	**91.0**	**4.8**	94.0	87.5	5.7
	200	88.8	83.0	10.9	92.2	62.5	6.4
	300	81.5	72.7	18.2	84.5	66.0	14.7
35	100	**97.0**	**98.1**	**3.1**	93.6	90.4	6.1
	200	86.7	78.9	13.1	86.2	78.9	13.7
	300	78.1	75.0	21.9	79.7	71.2	20.1
37	100	**98.8**	**97.0**	**1.2**	91.4	90.9	8.6
	200	88.2	87.9	11.9	87.5	81.8	12.5
	300	78.1	91.0	22.1	78.1	84.9	22.0

TABLE 6. ACC on IBRL dataset for three combination methods.

Node	Combination method: Proposed	Uniform	Local
1	**97.7**	94.1	95.3
2	**99.9**	89.1	99.3
33	94.1	**96.8**	93.7
35	**97.0**	96.9	89.3
37	**98.8**	93.6	94.2
Average	**97.5**	94.1	93.2

TABLE 7. DR on IBRL dataset for three combination methods.

Node	Combination method: Proposed	Uniform	Local
1	**96.4**	89.3	87.5
2	**97.7**	91.0	75.8
33	**91.0**	84.1	69.3
35	**98.1**	88.5	87.8
37	**97.0**	87.9	84.9
Average	**96.0**	88.1	81.1

TABLE 8. FAR on IBRL dataset for three combination methods.

Node	Combination method: Proposed	Uniform	Local
1	**2.3**	5.8	3.9
2	**0**	10.9	6.2
33	4.8	**2.7**	5.2
35	3.1	**2.9**	10.6
37	**1.2**	6.3	5.6
Average	**2.3**	5.7	6.3

The results shown in Table 6 illustrates that our combination method achieves the best ACC of 97.5% which better than the local and uniform combination methods. Similarly, Table 7 shows that an average of 96.0% DR is achieved with our combination method. And it outperforms the average DR of local and uniform combinations. In general, our framework with uniform and our combination method show an improvement over the local method. Among all three combination methods, our combination method shows the best performance in ACC and DR for our framework.

Finally, Table 8 shows the average of FAR of all combination methods. Our combination method has the FAR of 2.3% which outperforms other combination methods. This is due to our combination method considers the spatial correlation of sensor nodes and uses the actual distances among them to combine local detectors. Thus, it can effectively reduce the FAR. For the uniform combination method and the local detector, the difference of the FAR is slight. In a word, the proposed combination method effectively improves the ACC and the DR while significantly reduces the FAR.

V. CONCLUSION

In this paper, we proposed an isolation-based distributed outlier detection framework to detect outliers for wireless sensor networks. Our framework mainly solves three drawbacks of the exiting distributed outlier detection frameworks. The first one is the size growth of a local detection model in the training and detection phases which introduces extra computation and communication burden. In order to reduce resource consumption, we proposed a local detection model based on the iNNE algorithm. The second one is the poor performance of combination methods for local models. We introduced a new combination method based on the weighted voting method. The third one is the bad adaptability of dynamic changes in the environments for the existing frameworks. A self-adaptive algorithm based on a sliding window is developed in our local detection model. Extensive experiments on the ISSNIP dataset and the IBRL dataset show that the iNNE algorithm performs well for outlier detection. Though it has a similar training time with the iForest algorithm and the LOF algorithm, the detection performance of the iNNE is better than the other two algorithms. In addition, our framework outperforms a famous existing isolation-based framework in terms of ACC, DR, and FAR. Finally, we compared our combination method with local and uniform combination methods. The results show the proposed combination method effectively improves the ACC and the DR while significantly reduces the FAR. However, there are some shortages in our framework. The threshold and the size of the sliding window are predefined and fixed. In a real application, these values are hard to determine in advance and inappropriate values have a negative effect on the performance of the framework. Thus, a self-adaptive strategy which is able to adjust the threshold and window size of the detection framework is needed.

REFERENCES

[1] F. Kiani, "Animal behavior management by energy-efficient wireless sensor networks," *Comput. Electron. Agricult.*, vol. 151, pp. 478–484, Aug. 2018.

[2] P. Kumar, S. Kumari, V. Sharma, A. K. Sangaiah, J. Wei, and X. Li, "A certificateless aggregate signature scheme for healthcare wireless sensor network," *Sustain. Comput., Inform. Syst.*, vol. 18, pp. 80–89, Jun. 2018.

[3] A. J. Al-Mousawi and H. K. Al-Hassani, "A survey in wireless sensor network for explosives detection," *Comput. Elect. Eng.*, vol. 72, pp. 682–701, Nov. 2018.

[4] J. Lee, L. Kim, and T. Kwon, "FlexiCast: Energy-efficient software integrity checks to build secure industrial wireless active sensor networks," *IEEE Trans. Ind. Informat.*, vol. 12, no. 1, pp. 6–14, Feb. 2016.

[5] G. P. R. Filho, L. A. Villas, H. Freitas, A. Valejo, D. L. Guidoni, and J. Ueyama, "ResiDI: Towards a smarter smart home system for decision-making using wireless sensors and actuators," *Comput. Netw.*, vol. 135, pp. 54–69, Apr. 2018.

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

[7] Z. Yang, N. Meratnia, and P. Havinga, "An online outlier detection technique for wireless sensor networks using unsupervised quarter-sphere support vector machine," in *Proc. Int. Conf. Intell. Sensors, Sensor Netw. Inf. Process. (ISSNIP)*, Dec. 2008, pp. 151–156.

[8] A. Mahapatro and P. M. Khilar, "Fault diagnosis in wireless sensor networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 2000–2026, 4th Quart., 2013.

[9] W. Li, F. Bassi, D. Dardari, M. Kieffer, and G. Pasolini, "Defective sensor identification for WSNs involving generic local outlier detection tests," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 2, no. 1, pp. 29–48, Mar. 2016.

[10] Y. Zhang, N. Meratnia, and P. Havinga, "Outlier detection techniques for wireless sensor networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 12, no. 2, pp. 159–170, 2nd Quart., 2010.

[11] Y. Zhang, N. A. S. Hamm, N. Meratnia, A. Stein, M. van de Voort, and P. J. M. Havinga, "Statistics-based outlier detection for wireless sensor networks," *Int. J. Geograph. Inf. Sci.*, vol. 26, no. 8, pp. 1373–1392, 2012.

[12] Y. Yao, A. Sharma, L. Golubchik, and R. Govindan, "Online anomaly detection for sensor systems: A simple and efficient approach," *Perform. Eval.*, vol. 67, no. 11, pp. 1059–1075, 2010.

[13] H. Feng, L. Liang, and H. Lei, "Distributed outlier detection algorithm based on credibility feedback in wireless sensor networks," *IET Commun.*, vol. 11, no. 8, pp. 1291–1296, Jun. 2017.

[14] S. Rajasegarar, C. Leckie, M. Palaniswami, and J. C. Bezdek, "Distributed anomaly detection in wireless sensor networks," in *Proc. 10th IEEE Singapore Int. Conf. Commun. Syst. (ICCS)*, Oct./Nov. 2006, pp. 1–5.

[15] A. T. C. Andrade, C. Montez, R. Moraes, A. R. Pinto, F. Vasques, and G. L. da Silva, "Outlier detection using k-means clustering and lightweight methods for wireless sensor networks," in *Proc. 42nd Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, Oct. 2016, pp. 4683–4688.

[16] A. Abid, A. Kachouri, and A. Mahfoudhi, "Outlier detection for wireless sensor networks using density-based clustering approach," *IET Wireless Sensor Syst.*, vol. 7, no. 4, pp. 83–90, Aug. 2017.

[17] J. Gao, J. Wang, P. Zhong, and H. Wang, "On threshold-free error detection for industrial wireless sensor networks," *IEEE Trans. Ind. Informat.*, vol. 14, no. 5, pp. 2199–2209, May 2018.

[18] Z. Feng, J. Fu, D. Du, F. Li, and S. Sun, "A new approach of anomaly detection in wireless sensor networks using support vector data description," *Int. J. Distrib. Sensor Netw.*, vol. 13, no. 1, pp. 1120–1123, 2017.

[19] C. Titouna, M. Aliouat, and M. Gueroui, "Outlier detection approach using Bayes classifiers in wireless sensor networks," *Wireless Pers. Commun.*, vol. 85, no. 3, pp. 1009–1023, 2015.

[20] V. Chatzigiannakis, S. Papavassiliou, M. Grammatikou, and B. Maglaris, "Hierarchical anomaly detection in distributed large-scale sensor networks," in *Proc. 11th IEEE Symp. Comput. Commun. (ISCC)*, Jun. 2006, pp. 761–767.

[21] M. A. Rassam, M. A. Maarof, and A. Zainal, "A distributed anomaly detection model for wireless sensor networks based on the one-class principal component classifier," *Int. J. Sensor Netw.*, vol. 27, no. 3, pp. 200–214, 2018.

[22] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," *ACM SIGMOD Rec.*, vol. 29, no. 2, pp. 93–104, 2000.

[23] M. Xie, J. Hu, S. Han, and H.-H. Chen, "Scalable hypergrid k-NN-based online anomaly detection in wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 8, pp. 1661–1670, Aug. 2013.

[24] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation-based anomaly detection," *ACM Trans. Knowl. Discovery Data*, vol. 6, no. 1, p. 3, 2012.

[25] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proc. 8th IEEE Int. Conf. Data Mining*, Dec. 2008, pp. 413–422.

[26] Z.-G. Ding, D.-J. Du, and M.-R. Fei, "An isolation principle based distributed anomaly detection method in wireless sensor networks," *Int. J. Automat. Comput.*, vol. 12, no. 4, pp. 402–412, 2015.

[27] A. De Paola, S. Gaglio, G. Lo Re, F. Milazzo, and M. Ortolani, "Adaptive distributed outlier detection for WSNs," *IEEE Trans. Cybern.*, vol. 45, no. 5, pp. 902–913, May 2015.

[28] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," *ACM SIGOPS Oper. Syst. Rev.*, vol. 34, no. 5, pp. 93–104, Nov. 2000.

[29] L.-J. Zhao, T.-Y. Chai, and D.-C. Yuan, "Selective ensemble extreme learning machine modeling of effluent quality in wastewater treatment plants," *Int. J. Automat. Comput.*, vol. 9, no. 6, pp. 627–633, 2012.

[30] V. Chandola, A. Banerjee, and V. Kumar, "Outlier detection: A survey," *ACM Comput. Surv.*, vol. 14, Aug. 2007, p. 15.

[31] T. R. Bandaragoda, K. M. Ting, D. Albrecht, F. T. Liu, and J. R. Wells, "Efficient anomaly detection by isolation using nearest neighbour ensemble," in *Proc. IEEE Int. Conf. Data Mining Workshop (ICDMW)*, Dec. 2014, pp. 698–705.

[32] P. Swaruba and V. R. Ganesh, "Weighted voting based trust management for intrusion tolerance in heterogeneous wireless sensor networks," in *Proc. Int. Conf. Inf. Commun. Embedded Syst. (ICICES)*, Feb. 2014, pp. 1–7.

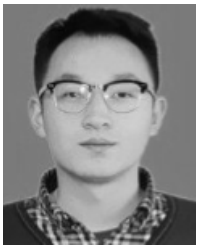
[33] M. Sun and H. Yang, "Gaussian process ensemble soft-sensor modeling based on improved bagging algorithm," *CIESC J.*, vol. 67, no. 4, pp. 1386–1391, 2016.

[34] S. Binsaeid, S. Asfour, S. Cho, and A. Onar, "Machine ensemble approach for simultaneous detection of transient and gradual abnormalities in end milling using multisensor fusion," *J. Mater. Process. Technol.*, vol. 209, no. 10, pp. 4728–4738, 2009.

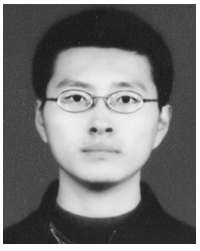
[35] S. Suthaharan, M. Alzahrani, S. Rajasegarar, C. Leckie, and M. Palaniswami, "Labelled data collection for anomaly detection in wireless sensor networks," in *Proc. 6th Int. Conf. Intell. Sensors, Sensor Netw. Inf. Process. (ISSNIP)*, Dec. 2010, pp. 269–274.

[36] P. Bodik, W. Hong, C. Guestrin, S. Madden, M. Paskin, and R. Thibaux, "Intel lab data," *Online Dataset*, to be published.

ZHONG-MIN WANG received the Ph.D. degree from the Beijing Institute of Technology, Beijing, China, in 2000. He is currently a Professor with the School of Computer Science and Technology, Xi'an University of Posts and Telecommunications. His current research interests include embedded intelligent perception, big data processing and application, and affective computing.



GUO-HAO SONG received the B.Eng. degree in communication engineering from the Xi'an University of Posts and Telecommunications, Xi'an, China, in 2017, where he is currently pursuing the master's degree in computer technology. His research interests include industrial big data, outlier detection, and wireless sensor networks.



CONG GAO received the Ph.D. degree in computer architecture from Xidian University, Xi'an, China, in 2015. He is currently an Assistant Professor with the School of Computer Science and Technology, Xi'an University of Posts and Telecommunications, Xi'an. His current research interests include data sensing and fusion, service computing, and wireless sensor networks.