Data anomaly detection and Data fusion based on Incremental Principal Component Analysis in Fog Computing

Xue-Yong Yu[1,2*] and Xin-Hui Guo[1,2*]

[1] Jiangsu Key Laboratory of Wireless Communications, Nanjing University of Posts and Telecommunications Nanjing 210003, China
[e-mail: yuxy@njupt.edu.cn, guoxinhui2014@163.com]
[2] Engineering Research Center of Health Service System Based on Ubiquitous Wireless Networks, Ministry of Education, Nanjing University of Posts and Telecommunications Nanjing 210003, China
* Corresponding author: Xin-Hui Guo and Xue-Yong Yu

Received April 2, 2020; revised June 22, 2020; revised July 8, 2020; accepted September 8, 2020; published October 31, 2020

Abstract

The intelligent agriculture monitoring is based on the perception and analysis of environmental data, which enables the monitoring of the production environment and the control of environmental regulation equipment. As the scale of the application continues to expand, a large amount of data will be generated from the perception layer and uploaded to the cloud service, which will bring challenges of insufficient bandwidth and processing capacity. A fog-based offline and real-time hybrid data analysis architecture was proposed in this paper, which combines offline and real-time analysis to enable real-time data processing on resource-constrained IoT devices. Furthermore, we propose a data process-ing algorithm based on the incremental principal component analysis, which can achieve data dimensionality reduction and update of principal components. We also introduce the concept of Squared Prediction Error (SPE) value and realize the abnormal detection of data through the combination of SPE value and data fusion algorithm. To ensure the accuracy and effectiveness of the algorithm, we design a regular-SPE hybrid model update strategy, which enables the principal component to be updated on demand when data anomalies are found. In addition, this strategy can significantly reduce resource consumption growth due to the data analysis architectures. Practical datasets-based simulations have confirmed that the proposed algorithm can perform data fusion and exception processing in real-time on resource-constrained devices; Our model update strategy can reduce the overall system resource consumption while ensuring the accuracy of the algorithm.

Keywords: Incremental Principal Component Analysis, Offline and real-time learning, Fog Computing, Data anomaly detection

This research was supported by the Natural Science Foundation of China under Grant 61871446, the Open Project of Wireless Communication Key Lab of Jiangsu Province (No. ZS002NY17013-2017WICOM03), and the Scientific Research Foundation of Nanjing University of Posts and Telecommunications (No. NY220047).

http://doi.org/10.3837/tiis.2020.10.004 **ISSN : 1976-7277**

1. Introduction

The Internet of Things provides ubiquitous services by connecting people and devices. As an important part of the IoT, the application for the agricultural monitoring is constantly improving. Most of the applications are based on wireless sensor networks, short-range wireless communication, and cloud computing [1]. With the expansion of production, the number of nodes will increase. The increasing number of nodes will raise the cost and will also generate large-scale, real-time, heterogeneous data, which, in turn, makes the application more complex. Thus, applications based on traditional cloud computing have problems with these aspects: limited bandwidth, high latency, slow response, and risk of service interruption. These problems will significantly affect the reliability and real-time of the application, and will represent an important challenge to the cloud in terms of access capabilities, storage capabilities, and processing capabilities.

There are significant characteristics of the agricultural environment data. For instance, agriculture environmental data is typically generated from wireless sensor nodes that are widely deployed in a wide range of production. Because of the large number and variety of nodes deployed, high-dimensional and large-scale data will be generated. In order to accurately capture the changes in the environment, nodes need to collect real-time data. As the changes in environmental indicators are relatively small in a short time, there is a lot of redundancy in the collected data. Uploading this data without compression can be a huge bandwidth consumption. In addition, the node may produce wrong data under the influence of energy status, communication ability and hardware. These data errors can be categorized into random errors and systematic errors. Data errors can significantly affect the reliability and effectiveness of data processing, especially in the early stage of data analysis application. Considering these characteristics of agricultural environmental data, it is necessary to realize real-time data compression and data anomaly detection in agricultural monitoring applications.

In order to solve the bandwidth bottleneck and delay problems in IoT applications based on traditional cloud services, relevant researches put forward the concept of fog computing [2,3]. Fog computing is a paravirtualized service computing architecture between cloud computing and personal computing. Building IoT applications based on fog computing has the following advantages:

1) **Make full use of the computing resources of tiny IoT devices**. These tiny devices are capable of communication and data processing. They cost less, but their computing resources are limited, so most of them are only used as data forwarding and storage devices in traditional IoT applications. Under the fog computing architecture, it is possible to optimize the traditional data processing algorithms and innovate the existing architecture, migrate applications to these tiny devices, thereby achieving full utilization of computing resources.

2) **Reduce bandwidth consumption**. Data processing applications based on fog computing can run much closer to where the data is generated so that most of the data can be processed locally without uploading to the cloud, which greatly reduces the bandwidth consumption.

3) **Provide data processing services with low latency, high stability, and well privacy protection**. The existing IoT applications rely on cloud service excessively. Cloud-based IoT applications have problems with high data transmission delay and privacy protection. Besides, the failure of cloud platforms directly affects the normal use of services, and the failure even causes service interruption. Whether the excessive delay or the risk of service interruption is unacceptable in medical application and safety production, the fog computing

architecture can significantly reduce the time delay and eliminate service outages, which protect the privacy of individuals.

Although fog computing has advantages showing above, the following problems still exist: The implementation of high-complexity algorithms is not supported on fog devices with limited resources, and it is difficult for the algorithm to guarantee the real-time processing. Though the existing relevant algorithms based on machine learning have realized anomaly detection and data fusion, they are not suitable for fog computing architecture due to their high complexity and requirements on computing resources. Moreover, most of these algorithms need to complete data analysis model training before they are applied to data analysis. Considering the real-time and large-scale characteristics of agricultural environmental data, it is difficult for these algorithms to deal with the real-time data of IoT and maintain high accuracy. An effective approach is to repeat the model training to obtain the latest model to ensure the accuracy of the algorithm, but it will cause a large waste of computing resources. Therefore, when applying the data processing algorithm in the fog computing architecture, it is necessary to consider the overall resource requirements of the algorithm, the real-time data processing capability, the resource consumption of the algorithm and whether the algorithm can achieve model update with low resource occupancy.

In this paper, we propose an offline and real-time data analysis architecture for fog computation analysis. The contributions of our work are summarized as follows:

1) The architecture realizes the real-time data processing by separating the model training and model usage. The offline system can perform model updating and training, while the real-time system relies on the model provided by the offline system to perform real-time data processing.

2) We construct an Incremental Principal Component Analysis (IPCA)-based data compression and anomaly detection algorithm to reduce the dimensionality and the scale of data upload. In addition, we utilize SPE [4] and the extracted PCs to detect abnormal data as the abnormality of the data will cause the defect of the SPE.

3) We propose a hybrid model update strategy to update the model at specific time intervals, and perform additional updates based on SPE to ensure the effectiveness of the model. The strategy supports the existing algorithms to perform real-time data processing, complete model updates with lower resources, reduce resource consumption, and ensure accuracy of the algorithm.

The rest of this paper is organized as follows: In Section II, some data anomaly detection and compression algorithms are surveyed and summarized. Section III elaborates on the details of the offline and real-time data analysis architecture based on fog computing, and Section IV elaborates on IPCA-based data compression and anomaly detection algorithms in detail. Our test results are discussed in Section V, while our conclusions are offered in Section VI.

2. Related Works on Data Compression and Anomaly Detection

To solve the problems in intelligence agriculture scenarios which caused by data anomalies and large-scale data uploads, related works have proposed some IoT data processing algorithms based on fog computing. Since the related work in this paper is based on research, a brief overview of this research is provided below.

1) **Clustering and classification algorithms based on machine learning**. P. Verma and S. K.Sood [5], an improved two-layer naive Bayes classifier is proposed, which captures

abnormal data based on the naive Bayes algorithm. A. Akbar et al. [6] researched data processing algorithms in traffic scenarios. The author integrated the single sensor data into events, and classified these events to obtain more accurate and complex event outputs. D. Borthakur et al. [7] used K-means clustering algorithm to extract data features on tiny computing devices, which divides the data into several clusters containing similar data features through continuous iteration. The algorithm uploads these features so that the scale of uploaded data will be reduced.

2) **Anomaly detection and data compression algorithms for sensor nodes**. These algorithms aim at detecting abnormal data collected by neighboring sensor nodes. S. Ji et al. [8] proposed an algorithm of node anomaly detection, which uses a weighted average method to detect the abnormality of the data uploaded by the node and use the result to determine the working status of the node. An idea of trust computing was proposed by G. Zhang et al. [9], which is to perform correlation analysis on data generated by multiple sensor nodes and upload trusted data, so as to remove redundant data and reduce bandwidth consumption. S. Ghiasi et al. [10] proposed an algorithm for supplementing the nearest neighbor data. The sensor nodes perform lightweight filtering on the collected data through preset thresholds. The data of multiple sensors will be analyzed uniformly to eliminate abnormal data. At the same time, in order to avoid the impact of data missing, the deleted abnormal data will be supplemented by the algorithm.

3) **Data compress and reduce**. To reduce the dimensionality of data, related works have been carried out [11-13]. The intuitive covariance-independent-free principal component analysis algorithm [14] (CCIPCA) does not need to calculate the covariance matrix during analysis, which can greatly reduce the consumption of computing resources and make the algorithm more complex and difficult to expand. However, the above algorithms have not considered the problem of model updating. T. Yu et al. [15] proposed an anomaly detection algorithm based on PCA and a method for the principal component update, which can update the principal component in real-time to achieve large-scale sensor data compression and anomaly detection. Hou, Ranran et al. [16] proposed a multi-parameter data anomaly detection algorithm based on IPCA. This algorithm uses an online model update strategy to ensure accuracy.

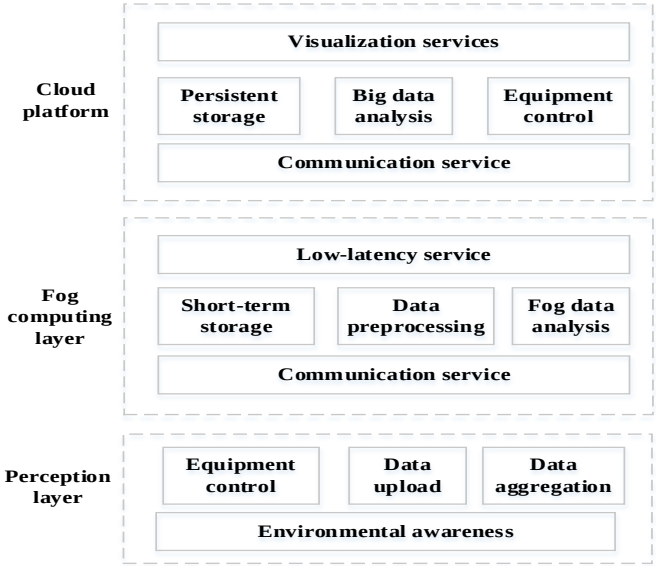

Fig. 1. IoT fog computing application architecture

In addition to research on data processing algorithms, this section also summarizes research on fog-based IoT applications and data processing architecture. M.Chiang and T. Zhang [17] elaborated on the challenges, advantages and development opportunities of fog computing, and introduced some typical fog-based applications. J. Moon et al. [18] proposes a framework and schemes for collaboration between the edge server and a cloud server. The framework aims to use the powerful information processing and search capabilities of cloud computing to enable edge computing devices to meet various performance requirements of heterogeneous wireless IoT networks. B. Tang et al. [19] introduced a fog-based multi-level data processing architecture in smart city pipeline monitoring applications. This architecture enables applications to cope with the challenges brought by the continuous growth of access devices by dispersing complex functions into different processing levels. Some fog-based medical data collection systems were proposed in [20,21], which realize real-time monitoring of the critical patients through real-time collection and analysis of medical data. Extensive related works were carried out to solve the problems of applying complex machine learning algorithms in resource-constrained scenarios [22,23], which implement data analysis on resource-constrained IoT nodes. In addition, related research also focuses on the data security and privacy issues of fog-based applications. In response to the privacy leakage problem of existing cloud service synchronization methods, Pooranian Z.et.al [24] proposed a RARE method that enables the cloud service to return a random response after receiving a deduplication request. This method can retain the advantages of existing synchronization methods and reduce the risk of user privacy leakage.

3. Offline and Real-Time Hybrid Data Analysis Architecture

Generally, fog-based IoT architecture can be implemented by adding a fog layer between the cloud platform and the perception layer, which is closer to the area where the data is generated. The application architecture is shown in **Fig. 1**.

In order to deploy data processing on fog computing, a data analysis architecture is proposed, as shown in **Fig. 2**. The architecture is implemented on one or more computing devices, which are located in the fog computing layer and have basic data exchange and processing capabilities. The architecture is mainly divided into two parts: offline analysis and real-time analysis. Offline analysis refers to offline model training without affecting real-time data processing, which can be deployed centrally or in a distributed manner with real-time analysis. Because offline analysis mainly completes model training in parallel, real-time analysis can be established based on the offline analysis. Also, offline analysis can obtain the latest model by repeating the model training or update the model by incremental analysis algorithms. At this time, the real-time analysis can use the latest model for data analysis, which ensures low latency and high accuracy, and also improves the efficiency of data analysis. It mainly relies on the model provided by offline analysis to handle real-time data. These processes include detecting abnormal and reducing the dimension of the data. Since there is no need for model training, real-time analysis can achieve the rapid analysis of small batches of data.

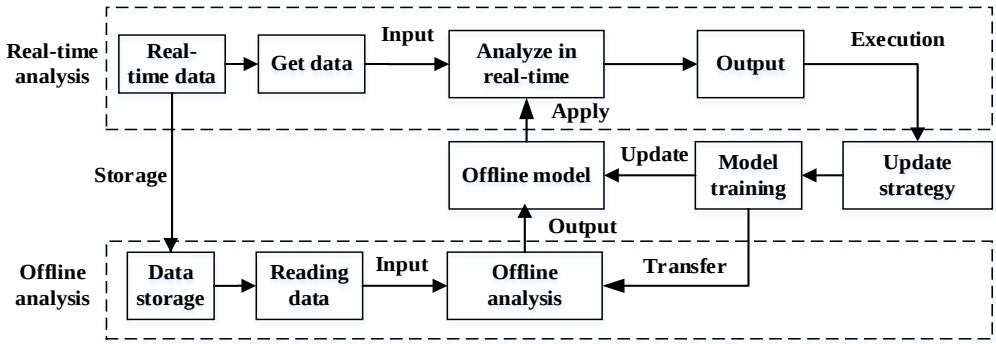

Fig. 2. Offline and real-time analysis architecture

In our proposed architecture, real-time data is uploaded by the perception layer and then handled by the real-time analysis system. This system will check for the latest analysis model, which is obtained based on the offline analysis system. Based on the acquired model, the real-time system conducts data analysis, and the obtained analysis results can be directly output to other systems to complete specific functions or be stored. The offline analysis system reads the latest data from the data storage system and conducts the training of the analysis model.

Although the architecture we proposed can realize the separation of model training and model using, perform real-time data processing and realize model updating, and reduce the execution time of the algorithm, but this architecture requires the deployment of multiple devices for parallel processing and the exchange of models and data, so the overall cost of the system will inevitably increase. The cost of the system consists of two parts: the computational expense and communication cost. The computational expense includes equipment cost and algorithm execution cost, and the communication cost mainly comes from the information transmission of the equipment. Part of the cost is necessary to maintain normal system operation. For this reason, when the number of devices deployed in the system increases, the system cost also increases. In addition, we need to confirm the impact of offline and real-time analysis on system costs. In our proposed architecture, offline analysis mainly performs model training, and communication with real-time analysis only involves data reception and a small amount of model transmission. In contrast, the real-time analysis does not need to perform model training, but it needs to send a large amount of real-time data and model update requests to offline analysis. Therefore, when considering the overall resource of the architecture, the increase in communication and computing costs is inevitable. For this reason, we mainly focus on the optimization of the computational expense of offline analysis and the optimization of the communication cost of real-time analysis.

In order to reduce the cost of the system, it is necessary to avoid unnecessary model updates, while reducing the frequency of data and model update requests. To this end, we mainly focus on the model update strategies. Reasonable update strategies can reduce the number of model training and information exchange between devices, reduce the system's computational expense and communication cost, and make the system adapt to different application scenarios. Some basic model update strategies include regular strategies and real-time strategies:1) **Update the model in real-time**. In the real-time update strategy, offline model training is performed when new data arrives. This strategy requires repeated execution of the algorithm, which consumes a lot of computing resources. In most cases, the

updated model does not change much from the previous model. Therefore, this strategy is not applicable in most cases.2) **Update the model regularly**. The regular update strategy is to perform offline model training at certain intervals. The update interval of the model can be adjusted according to the real-time and accuracy requirements of the application, so it has high adaptability. However, the disadvantages of the regular update strategy are also obvious. When the amount of data processed is small, and the data changes greatly in a short period, if the model cannot be updated, the abnormal data may be missed or misjudged.

In addition to the above update strategies, other different update strategies can be applied to the system. After the system confirms the model update strategy, it decides whether to training the new analysis model according to the specified update strategy. If the update is eligible, the offline analysis system will be called, and the new analysis model will be updated to the real-time analysis system. Therefore, the real-time system can use the updated model for analysis. Compared with integrating model training and data analysis functions into one system, the proposed data analysis architecture has the following characteristics: The architecture supports the distribution of different computing tasks to different nodes or the handing of complex computing tasks to cloud services, which can solve the problem of the insufficient computing capacity of fog nodes to a certain extent and enable complex algorithms to be applied in fog computing scenarios; By applying different model update strategies, the architecture can adapt to different scenarios and requirements and reduce the overall resource consumption of the algorithm.

4. Data compression and anomaly detection algorithm based on incremental principal component analysis

In this section, we mainly elaborate on the proposed data processing algorithm., which is divided into two parts: data compression and data anomaly detection. In addition, we also introduce the specific implementation of the algorithm in the offline and real-time hybrid data analysis architecture.

4.1 Data compression algorithm

In this section, we first introduce the PCA algorithm briefly, which is the basis of the IPCA algorithm. Suppose the data is stored in the form of a matrix, where matrix rows represent data records, and matrix columns represent data features.

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1k} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nk} \end{bmatrix} = (X_1, X_2, \ldots . X_k) \tag{1}$$

Considering that there are differences in the value range of different dimensions of data, and the algorithm requires the data conform to the characteristics of the normal distribution, the Z-score normalization method is used to normalize the data.

$$x^* = \frac{x-\mu}{\sigma} \tag{2}$$

The PCA calculates the covariance matrix Z and performs eigenvalue decomposition to obtain a vector matrix containing the PCs (Principal Components).

$$Z = P\Lambda P^T \tag{3}$$

Where P contains d eigenvectors corresponding to the eigenvalues, d represents the number of vectors in the covariance matrix, and Λ is the eigenvalue matrix. Suppose the eigenvectors $P(u_1, u_2, \ldots u_d)$ corresponding to d largest eigenvalues of Λ are obtained, we get m PCs $(pc_1, pc_2, pc_3, \ldots pc_m)$ from P. After the PCs are extracted, the PCA algorithm can realize the dimensionality reduction of the data.

In the previous part, we elaborate on the data compression algorithm based on the PCA. Compared with the PCA, IPCA can complete model updates based on existing models and the small batch of new input data and requires fewer resources. We assume that the analysis model of IPCA is $\omega = (\mu, \sigma, P, \Lambda, N)$, where μ represents the mean of the data, σ represents the standard deviation, and N is the number of current data. Before updating the PCs, the mean and standard deviation needs to be updated. The parameter update method is expressed as

$$\mu' = \frac{1}{N+1}\mu + \frac{1}{N+1}X_{N+1} \tag{4}$$

$$\sigma = \frac{N}{N+1}\sum_{i=1}^{N}(x_i - \mu')^2 + \frac{1}{N+1}(x_{N+1} - \mu')^2 \tag{5}$$

X_{N+1}represents the current input data. According to (4) and (5), the mean and standard deviation can be updated when new data arrives. After the mean and standard deviation are updated, the IPCA algorithm can update the existing principal component. When new data X_{N+1} is input [25], the residual vector r can be expressed as

$$r = (x_{N+1} - \mu') - PP^T(x_{N+1} - \mu') \tag{6}$$

The eigenvalue decomposition of the updated covariance matrix Z is required to obtain the updated PCs. When new data is added to the data matrix, the new covariance matrix Z' is expressed as

$$Z' = \begin{bmatrix} Z & x_{N+1} \\ 0 & ||r|| \end{bmatrix} \tag{7}$$

Based on the eigenvalue decomposition of the new covariance matrix Z', the result can be used to update the original PCs. The dimension reduction of data can be realized based on the PCs updated.

4.2 Data anomaly detection algorithm

After extracting the PCs and compressing the data, we elaborate on the data anomaly detection algorithm based on the SPE value and the PCs. The SPE value refers to the square of the norm between the original data matrix and the reconstructed data matrix, which is similar to the concept of residual in mathematical statistics. Data refactoring takes the following approach.

$$X_{re} = X * P * P^T \tag{8}$$

After data reconstruction, the SPE value is calculated as follows

$$SPE(t) = ||X - X_{re}||_2^2 \tag{9}$$

The above formula is used to calculate the SPE value of a single data. In the analysis of batch data, we generally take the mean value of the SPE value calculated from all data. When the data is approximately normal distribution, based on the relevant work in [15], we use the pauta criterion to detect data anomalies. The pauta criterion holds that data are reliable if the absolute value of the difference between the measured data and the estimated data is less than three times the standard deviation of the measured data. If the standard deviation of the data exceeds three times the standard value, the data can be considered abnormal; otherwise, the data can be considered in the normal range. The pauta criterion is expressed as follows:

$$|SPE(t) - \mu_{SPE}| > \varepsilon_a * \sigma_{SPE} \tag{10}$$

SPE (t) represents the SPE value of the current data, μ_{SPE} represents the average value of all SPEs, $\varepsilon_a$is 3, and $\sigma_{SPE}$is the standard deviation of SPE values. Based on the above process, we can complete data compression and anomaly detection.

Considering that the existing update strategy for the model cannot solve the problem of high resource occupancy caused by the data analysis architecture, in order to avoid the impact of abnormal data on the accuracy of data analysis, we design a regular-SPE hybrid model update strategy to achieve the model update on demand and reduce the overall resource consumption. Different from the model update strategy based on the SPE proposed in [27], the regular strategy can ensure that the model keeps updating steadily, thereby ensuring the accuracy of the algorithm model in the early stage of application. At the same time, the SPE value can also solve the problem that the regular update strategy cannot be updated immediately and the update interval is difficult to set. The update strategy is as follows

$$\begin{cases} SPE > \eta & update\ immediately \\ SPE < \eta & update\ regularly \end{cases} \tag{11}$$

where η represents the threshold of model updates, SPE represents the distance between the current data point and the projection plane of the principal component. When the SPE value of the data does not exceed the threshold value, it is considered that the influence of the current data on the PCs is within the controllable range, and the PCs do not need to be adjusted. Therefore, the regular update strategy can be adopted to consider the effectiveness of the algorithm and the consumption of resources. In addition, when the SPE value of the data exceeds the threshold value, it indicates that the data may have a great impact on the principal component, and the model update is required to ensure the effectiveness of the algorithm. This strategy fully takes into account the problem that the algorithm is greatly affected by abnormal data in the initial stage of analysis, and realizes the updating of PCs on demand. Compared with the regular updating strategy, our strategy can capture the changes of data.

4.3 Implementation of the algorithm

Combined with the proposed model update strategy and data analysis architecture, we implement the data compression and anomaly detection algorithm based on the IPCA. The algorithm consists of two parts: one is offline analysis, the other is real-time analysis. The flow of the algorithm is shown in **Fig. 3**.

Before data analysis, all data needs to be normalized. During initialization, the offline analysis system will complete the training of the initial model, which is mainly based on

algorithm 1 and the dataset stored in the system. After the initial model training, the offline analysis will wait for the model update request of the real-time analysis. Then, the offline analysis system retrieves the latest input data and updates the model and parameters after receiving the update request. Finally, the latest model will be returned to the real-time analysis system. When running for the first time, the real-time analysis system will invoke the initial offline model. Then, it performs real-time data compression to calculate SPE values according to algorithm 2 and performs abnormal detection of data. At this time, the system will store normal data, and when the abnormal data is detected, the system will selectively call the offline system according to the preset update strategy and data anomaly detection results.

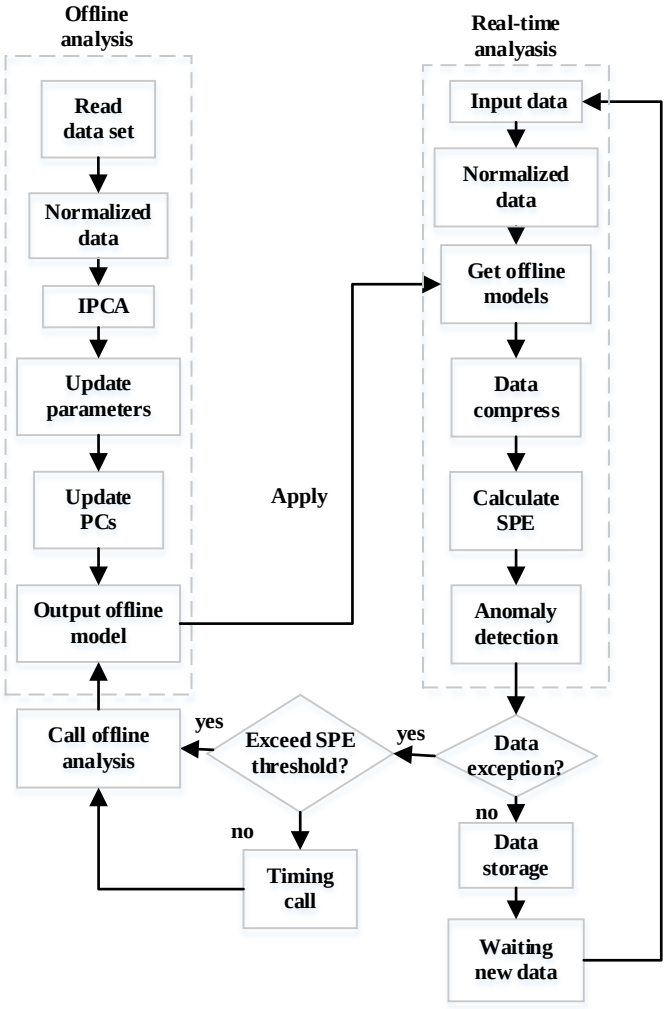

Fig. 3. Data compression and anomaly detection algorithm flow

5. Performance Evaluation

This section mainly evaluates the performance of the proposed data compression and anomaly detection algorithm from two aspects: the anomaly detection capability of the algorithm and the loss after data compression and reconstruction.

Algorithm1：IPCA-Based Offline Data Analysis algorithm

Initialization：read data from dataset normalized $X\sim N(0,1)$
initialize mean μ, standard deviation σ
Calculate covariance matrix
$$Z=\frac{1}{m}XX^T$$
Eigenvalue decomposition Z matrix to obtain matrix P
$$Z=P\Lambda P^T$$
Get the initial principal component model pcs
If get update request from real-time analysis
Get the latest data entry X_{new} and normalized $X\sim N(0,1)$
For x in X_{new}
Update mean value and standard deviation
$$\mu'=\frac{1}{N+1}\mu+\frac{1}{N+1}x$$
$$\sigma=\frac{N}{N+1}\sum_{i=1}^{N}(x_i-\mu')^2+\frac{1}{N+1}(x-\mu')^2$$
Update Covariance matrix Z'
Eigenvalue decomposition Z'
Get new principal component model pcs

Algorithm2：Data anomaly detection algorithm based on SPE value and principal component model

Initialization： Extract the latest offline principal component model pcs
Get initial $\overline{SPE}$
Get new input data matrix X ,normalized $X\sim N(0,1)$
Data reconstruction $X_{re}=X*P*P^T$
Calculate SPE matrix $SPE=||X-x_{re}||_2^2$
Calculate $\overline{SPE}$ value μ_{SPE}
For x(t) in X
Data reconstruction $x_{re}=x(t)*P*P^T$
Calculate SPE(t) value $SPE(t)=||x(t)-x_{re}||_2^2$
Data anomaly determination
if $|SPE(t)-\mu_{SPE}|>\varepsilon_a*\sigma_{SPE}$:
Data is abnormal
Call offline analysis to update the model immediately
X matrix output to offline system
else
Data is normal
Regularly call offline analysis to update the model
Data output to offline system to store

To facilitate the evaluation of our proposed algorithm, we test it on the IntelLab dataset [26]. The dataset mainly includes data generated by 54 sensor nodes within two months, with an interval of 31 seconds. The data types include temperature, humidity, light, voltage, which are widely concerned in intelligent agriculture applications. Based on the Anaconda data analysis integration environment and the sklearn machine learning library, our tests evaluate

the performance of the algorithm and the effectiveness of the model update strategy on both PC and raspberry PI devices.

In order to ensure reliability, the amount of data to be analyzed is about 50000.These data are from the same sensor node, whose node number is 1. The algorithm selects three dimensions of temperature, humidity and illumination in the dataset to conduct data dimensionality reduction, and the target dimension is 2. The amount of data read per time set by the IPCA algorithm is 20, and the threshold value of SPE value for anomaly detection is set to 3 times the standard deviation of the reconstructed data matrix. The data processing results of this algorithm are shown in figure **Fig. 4**.

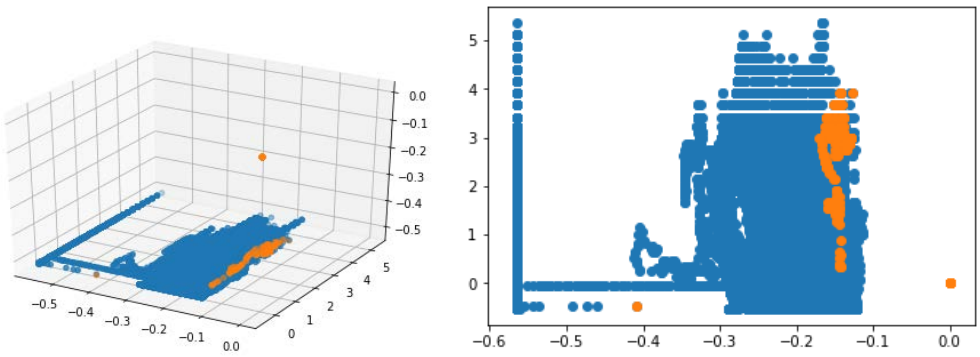

Fig. 4. Data processing results

Fig. 4 shows the initial dataset (left) and the dataset after the dimensionality reduction (right), the abnormal data in the dataset are also marked prominently. In addition, in order to compare the difference of the data processing results obtained by the IPCA-Based algorithm and the PCA-Based algorithm, we test the two algorithms based on the dataset of different sensor nodes. The test results show that the average difference between the processing results obtained by the two algorithms is less than 2% for both large and small datasets. Therefore, the IPCA-Based algorithm has similar compression and reconstruction capabilities as the PCA-Based algorithm.

To test the anomaly detection capabilities of the algorithm, we add some abnormal data to the dataset. To avoid the influence of data value and data distribution on the test results, we set the data to 0 and distribute data evenly in the dataset. In addition, we mainly use the TPR (True Positive Rate) and FPR (False Positive Rate) as evaluation criteria to evaluate the anomaly detection capability of the algorithm [27]. Then we test the algorithm on multiple data sets and record the results, the average of which is aggregated in **Table. 1**.

Table 1. Comparison of Anomaly Detection Capabilities of Different Algorithms

Algorithms	TPR	PFR
PCA	1	0.19
IPCA	1	0.17

Our statistics on outliers do not include the outliers that already exist in the dataset. It can be seen from the table that both the IPCA-Based algorithm and the PCA-Based algorithm can detect all the abnormal data which are set in the dataset in multiple tests, with a TPR rate of 100%. Since the model training and updating process is continuous, when the trend of data changes, some data that are obviously normal in the complete dataset may be misjudged

as abnormal data. This problem is particularly significant in incremental algorithms. We compare the results of batch training and incremental training to roughly determine the part of the data. In addition, different dimensions in the data set have different weights when performing dimensionality reduction. Some abnormal data with low weight dimensions may not cause dramatic changes in the SPE value of the data group where the abnormal data is located, which will make our method unable to find some data abnormalities. We confirm this part of the data by secondary judgment, which is mainly achieved by calculating the ratio between the SPE value of a certain dimension data and the overall SPE value of the data. The above uncertain data anomalies are included in the statistics of relevant parameters From the PFR statistics in the table, and it can be seen that the PFR rate of the PCA-Based algorithm and the IPCA-Based algorithm is similar, and the difference is less than 3%.

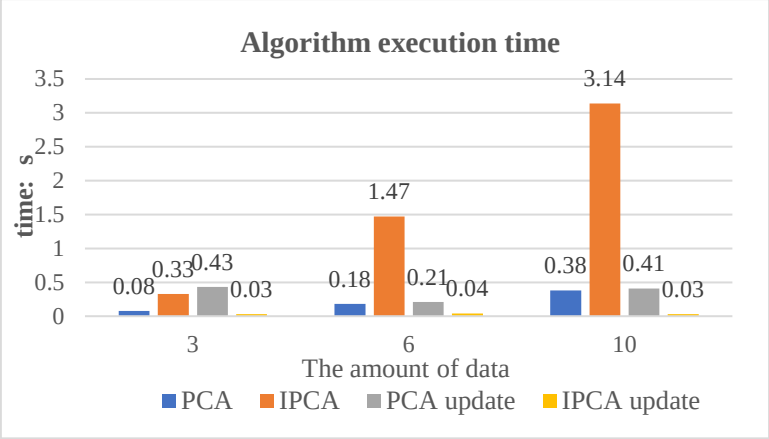

Fig. 5. Algorithm execution time statistics

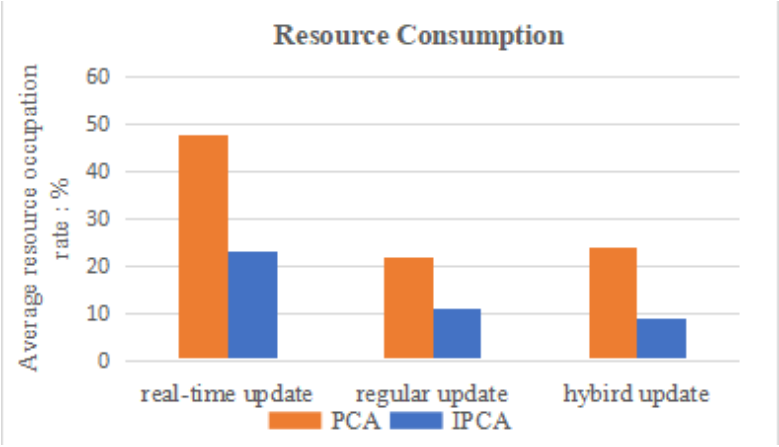

Fig. 6. Performance comparison of different update strategies

The algorithm we proposed requires the training of the initial model, it is necessary to count the execution time for different algorithms to complete the initial model training under different data scales. In addition, the time of model updating by different algorithms is also counted. In order to facilitate the comparison, we set 1000 pieces of data to be input each time for model updating. The statistical results are shown in **Fig. 5**.

Fig. 5 contains the statistical results of the time required for the initial model training and model update by different algorithms. As shown in **Fig. 5**, with the expansion of data scale, the training time gap between the IPCA-Based algorithm and the PCA-Based algorithm continues to increase. When the size of the dataset is 100,000, the training time of the IPCA-Based algorithm is about 8 times the PCA-Based algorithm. Although the initial model training time of the IPCA-Based algorithm is much higher than that of the PCA-Based algorithm, the time required for the IPCA-Based algorithm to update the PCs is much less than the PCA-Based algorithm. In our test, when we input 1000 new data, the IPCA-Based algorithm can update the model in 0.03 seconds, which much less than 0.4 seconds of the PCA-Based algorithm. These results indicate that the IPCA-Based algorithm has higher real-time performance, and it has significant advantages in data processing scenarios where model updating is needed frequently.

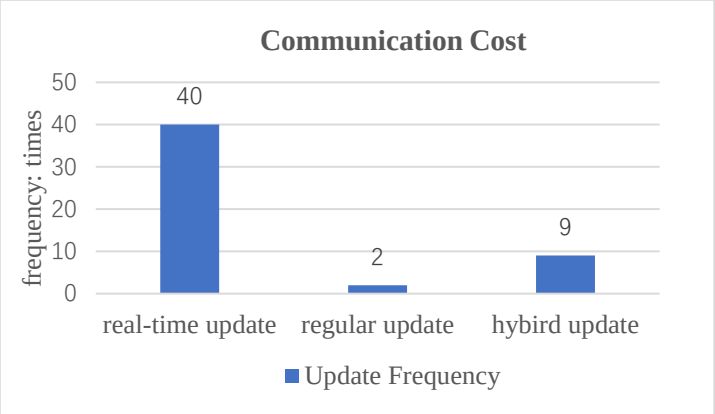

Fig. 7. Communication cost of different update strategies

After analyzing the execution time of the algorithm, we also analyzed the computational expense and communication cost of different algorithms under different update strategies. We focus on the CPU and memory usage of the algorithm when it runs on raspberry PI devices and evaluate the performance of both as a whole. In the evaluation of communication cost, considering that the amount of data to be sent by different algorithms and strategies in the real-time analysis system is almost the same, and the model update only needs to send a small amount of data, for this reason we use the number of data transmissions to estimate the communication cost. We mainly test three model update strategies. The model update time of the regular strategy is set to 30 minutes, and the data size of the initial training is set to 60,000. The update threshold η is set to 1. **Fig. 6** shows the resource consumption of the PCA-Based algorithm and the IPCA-Based algorithm in model updating under different update strategies. **Fig. 7** shows the communication cost in model updating under different update strategies.

As shown in **Fig. 6**, compared with the PCA-Based algorithm, the IPCA-Based algorithm reduces the overall resource consumption by about 30%. In the comparison of the resource consumption of the update strategy, the hybrid strategy is about 35% less than the real-time strategy and 10% more than the regular strategy. The reason is that the hybrid strategy combines the regular strategy and the detection of abnormal data. When there are more anomalies in the input data, the hybrid strategy is required to update the model more frequently, which will lead to an increase in resource consumption. When there is no outlier in the input data, the performance of the hybrid strategy is similar to the regular strategy,

because the data outlier does not need to be considered. As shown in **Fig. 7**, the transmission frequency of the real-time strategy is the highest, while the regular strategy is the least. The transmission frequ- ency of the hybrid strategy is much smaller than the real-time strategy, and slightly larger than the regular strategy. In summary, in the scenario of real-time data processing, we can tell the hybrid update strategy can significantly reduce the computational expense compared to other traditional update strategies, and also has certain advantages in communication cost.

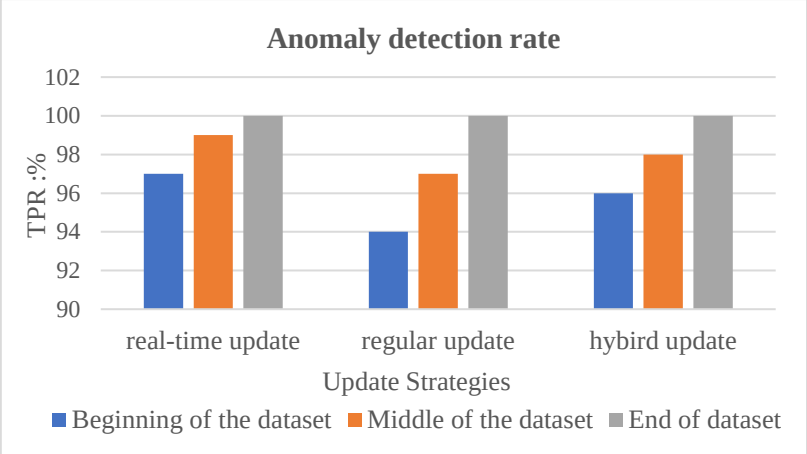

Fig. 8. Anomaly detection rate under different update strategies

After analyzing the resource consumption of different update strategies, it is also necessary to analyze the anomaly detection performance of different update strategies. In this section, we constructed multiple data distribution states to complete our test. We use 10,000 data points for initial model training. We take 1,000 data points from the dataset for input each time while we set the total number of samples to 50,000. The test result statistics are shown in **Fig. 8**.

As shown in **Fig. 8**, when abnormal data are distributed at the end of the dataset, and the overall data scale is large, no matter which update strategy has a relatively high detection rate, which is mostly close to 100%. When the abnormal data is distributed in the initial part of the dataset, the real-time strategy is less affected, and the TPR value is about 97%. However, due to the lack of sufficient data, it is impossible to ensure that all the abnormal data are detected if the initial data is small. Compared with the other two algorithms, the regular strategy is greatly affected by the data distribution. Especially when there are a lot of abnormal data in the initial stage of data analysis, the average TPR is 94%, which is far less than the real-time strategy and the hybrid strategy. The hybrid strategy is less affected by abnormal data distribution and has higher accuracy. In a variety of data distribution conditions, its TPR is higher than 96%, which is also higher than the regular strategy. From the previous analysis, it can be seen that compared with the real-time strategy and the regular strategy, the hybrid strategy can detect data anomalies with lower resource consumption and adapt to different abnormal data distribution.

6. Conclusion

In this paper, we proposed an offline and real-time hybrid data analysis architecture based on fog computing. By combining offline and real-time analysis, the architecture enables traditional algorithms to perform real-time data processing. Based on this architecture, we proposed an IPCA-Based data compression and anomaly detection algorithm, which can efficiently handle large-scale data with low memory costs and fewer resource requirements. In addition, in order to enable the algorithm to process real-time data, we designed a principal component model update strategy. This strategy combines the regular update strategy and the data anomaly detection algorithm to update the model on-demand and to ensure the accuracy of the algorithm and reduce resource consumption. We implemented the algorithm on PC and Raspberry PI devices and tested the algorithm based on the public dataset. Relevant test results showed that the proposed model update strategy and data analysis algorithm could adapt to a variety of scenarios and obtain higher efficiency and lower resource consumption.

References

[1] Wang Yi, “Analysis on the Application Mode of Cloud Computing in the Internet of Things,” *Telecommunications Science*, 27(12), 26-30, 2011. Article (CrossRef Link)

[2] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog computing and its role in the Internet of Things,” in *Proc. of 1st Ed. MCC Workshop Mobile Cloud Comput. (MCC), New York, NY, USA*, pp. 13–16, 2012. Article (CrossRef Link)

[3] M. Taneja and A. Davy, “Resource aware placement of IoT application modules in fog-cloud computing paradigm,” in *Proc. of IFIP/IEEE Symp. Integr. Netw. Service Manage. (IM)*, pp. 1222–1228, 2017. Article (CrossRef Link)

[4] V. Chatzigiannakis and S. Papavassiliou, "Diagnosing Anomalies and Identifying Faulty Nodes in Sensor Networks," *IEEE Sensors Journal*, vol. 7, no. 5, pp. 637-645, May 2007. Article (CrossRef Link)

[5] P. Verma and S. K. Sood, "Fog Assisted-IoT Enabled Patient Health Monitoring in Smart Homes," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1789-1796, June 2018. Article (CrossRef Link)

[6] A. Akbar et al., "Real-Time Probabilistic Data Fusion for Large-Scale IoT Applications," *IEEE Access*, vol. 6, pp. 10015-10027, 2018. Article (CrossRef Link)

[7] D. Borthakur, H. Dubey, N. Constant, L. Mahler and K. Mankodiya, "Smart fog: Fog computing framework for unsupervised clustering analytics in wearable Internet of Things," in *Proc. of 2017 IEEE Global Conference on Signal and Information Processing (Global SIP), Montreal, QC*, pp. 472-476, 2017. Article (CrossRef Link)

[8] S. Ji, S. Yuan, T. Ma and C. Tan, "Distributed Fault Detection for Wireless Sensor Based on Weighted Average," in *Proc. of 2010 Second International Conference on Networks Security, Wireless Communications and Trusted Computing, Wuhan, Hubei*, pp. 57-60, 2010. Article (CrossRef Link)

[9] G. Zhang and R. Li, "Fog computing architecture-based data acquisition for WSN applications," *China Communications*, vol. 14, no. 11, pp. 69-81, Nov. 2017. Article (CrossRef Link)

[10] S. Ghiasi, A. Srivastava, X. Yang, and Sarrafzadeh, “Optimal energy aware clustering in sensor networks,” *Sensors*, vol. 2, no. 7, pp. 258– 269, 2002. Article (CrossRef Link)

[11] V. Chatzigiannakis and S. Papavassiliou, "Diagnosing Anomalies and Identifying Faulty Nodes in Sensor Networks," *IEEE Sensors Journal*, vol. 7, no. 5, pp. 637-645, May 2007. Article (CrossRef Link)

[12] P. Poekaew and P. Champrasert, "Adaptive-PCA: An event-based data aggregation using principal component analysis for WSNs," in *Proc. of 2015 International Conference on Smart Sensors and Application (ICSSA), Kuala Lumpur*, pp. 50-55, 2015. Article (CrossRef Link)
[13] A. Rooshenas, H. R. Rabiee, A. Movaghar and M. Y. Naderi, "Reducing the data transmission in Wireless Sensor Networks using the Principal Component Analysis," in *Proc. of 2010 Sixth International Conference on Intelligent Sensors, Sensor Networks and Information Processing, Brisbane, QLD*, pp. 133-138, 2010. Article (CrossRef Link)
[14] Weng J, Zhang Y, Hwang W S., "Candid covariance-free incremental principal component analysis," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 25(8), 1034-1040, 2003. Article (CrossRef Link)
[15] T. Yu, X. Wang and A. Shami, "Recursive Principal Component Analysis-Based Data Outlier Detection and Sensor Data Aggregation in IoT Systems," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 2207-2216, Dec. 2017. Article (CrossRef Link)
[16] Hou, Ranran, et al., "Incremental PCA based online model updating for multivariate process monitoring," in *Proc. of the 10th World Congress on Intelligent Control and Automation IEEE*, 2012. Article (CrossRef Link)
[17] M. Chiang and T. Zhang, "Fog and IoT: An Overview of Research Opportunities," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 854-864, Dec. 2016. Article (CrossRef Link)
[18] J. Moon, S. Cho, S. Kum and S. Lee, "Cloud-Edge Collaboration Framework for IoT data analytics," in *Proc. of 2018 International Conference on Information and Communication Technology Convergence (ICTC), Jeju*, pp. 1414-1416, 2018. Article (CrossRef Link)
[19] B. Tang et al., "Incorporating Intelligence in Fog Computing for Big Data Analysis in Smart Cities," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 5, pp. 2140-2150, Oct. 2017. Article (CrossRef Link)
[20] T. N. Gia, M. Jiang, A. Rahmani, T. Westerlund, P. Liljeberg and H. Tenhunen, "Fog Computing in Healthcare Internet of Things: A Case Study on ECG Feature Extraction," in *Proc. of 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, Liverpool*, pp. 356-363, 2015. Article (CrossRef Link)
[21] O. Akrivopoulos, I. Chatzigiannakis, C. Tselios and A. Antoniou, "On the Deployment of Healthcare Applications over Fog Computing Infrastructure," in *Proc. of 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, Turin, pp. 288-293, 2017. Article (CrossRef Link)
[22] S. Dey and A. Mukherjee, "Implementing Deep Learning and Inferencing on Fog and Edge Computing Systems," in *Proc. of 2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), Athens*, pp. 818-823, 2018. Article (CrossRef Link)
[23] T. Tuor, S. Wang, T. Salonidis, B. J. Ko and K. K. Leung, "Demo abstract: Distributed machine learning at resource-limited edge nodes," in *Proc. of IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Honolulu, HI*, pp. 1-2, 2018. Article (CrossRef Link)
[24] Pooranian Z, Conti M, Yu C M, "RARE: Defeating Side Channels based on Data-Deduplication in Cloud Storage," in *Proc. of Infocom Workshops Ccsna*, 2018. Article (CrossRef Link)
[25] B. Li, D. Yao and Y. Qian, "Incremental principal component analysis method on online network anomaly detection," in *Proc. of 2013 International Conference on Information and Network Security (ICINS 2013), Beijing*, pp. 1-6, 2013. Article (CrossRef Link)
[26] M. Samuel, "Intel Lab Data," *Massachusetts Avenue, Boston, MA, USA, and MIT, Cambridge, U.K.*, pp. 12–31, 2012. Article (CrossRef Link)
[27] Zhou Zhihua, *Machine Learning*, Tsinghua University Press, Beijing, pp. 33-34, 2016. Article (CrossRef Link)

Xue-Yong Yu was born in Jiangxi, China, in 1979. He received the Ph.D. degree in electromagnetic field and microwave technology from the Nanjing University of Posts and Telecommunications, Nanjing, China, in 2016, where he is currently an Associate Professor. His current research interests include the Internet of Thing (IoT), mobile edge computing, and radio resource management on heterogeneous wireless networks.

in-Hui Guo received the B. S degree in Optoelectronic Information Science and Engineering from Tonga College of Nanjing University of Posts and Telecommunications (NYTDC), in 2017.He is currently pursuing the master' degree with the Nanjing University of Posts and Telecommunications (NUPT), where he involved in IoT and fog computing.