Article

Research on Fault Detection by Flow Sequence Algorithm with Deep Learning Model for Industrial Internet of Things

Dongfeng Lei , Liang Zhao , Dengfeng Chen *

Posted Date: 1 March 2024

doi: 10.20944/preprints202403.0054.v1

Keywords: Industrial Internet of Things; Deep learning; Subsequence classification; Fault diagnosis

Article

Research on Fault Detection by Flow Sequence Algorithm with Deep Learning Model for Industrial Internet of Things

Dongfeng Lei, Liang Zhao and Dengfeng Chen *

College of Information and Control Engineering, Xi 'an university of Architecture and Technology, Xi'an 710055, China; lllddddfffldf@163.com (L.D.F.); zhaoliang@xauat.edu.cn (Z.L.); chdengf@xauat.edu.cn (C.D.F.)
* Correspondence: lllddddfffldf@163.com

Abstract: Classifying the flow subsequences of sensor networks is an effective method for fault detection in the Industrial IoT. Traditional fault detection algorithms, detect faults using a single sensor input and do not pay enough attention to factors such as electromagnetic interference, network delay, and sensor sample delay. This results in several maladjustment problems. This paper proposes a fault detection algorithm called SSGBUL-IKNN based on the flow sequence of the sensor network to achieve higher fault detection accuracy. Firstly, we built a network module based on unsupervised learning to encode the flow sequence of the sensor network. Then, the specific fault type was determined through an improved KNN classification algorithm based on the indicated distance of the source from the integrated coding sequences of the sensor network. The results obtained from tests on three Industrial IoT datasets of a sewage treatment plant shows that the accuracy of SSGBUL-IKNN algorithm exceeds 90%, which is significantly higher than the accuracies of the DTW and TSF algorithms. The proposed algorithm reaches the classification requirements for fault detection in an Industrial IoT.

Keywords: Industrial Internet of Things; deep learning; subsequence classification; fault detection

1. Introduction

In the Industry 4.0 era, the Industrial Internet of Things (IIoT) has become increasingly important to industrial production [1,2]. Sensors are data collection and perception nodes in IIoT systems. The data collected by sensors provides real-time and accurate information foundation for IIoT systems, promoting the development of IIoT technology and the diversification of application scenarios. In the IIoT system, timely and accurate diagnosis of faults is very important, which can help quickly identify the problem and take effective measures to repair it.

Sensors sample factory field information and are also responsible for executing commands from the control center to adjust the behaviors of various equipment. Different kinds of sensors hold fixed numbers of digital requestors and analog registers. Some of these sensors are used to provide monitoring information, such as flow meters, PH concentration meters, and so on. Others are used to provide configuration parameters for engines, for example, frequency converter, water pumps, fans, and so on. The remote I/O module samples these sensors' data through a related interface included in the sensor, and sets up a standard field bus interface for the gateway. The gateway communicates with the remote I/O in a fixed period, and then translates the sensor data to the Message Queuing Telemetry Transport (MQTT) format before finally sending it to the cloud server.

Various faults can occur during the operation of the IIoT [3], such as sensor disconnection, offline remote I/O modules, illegal system access, cyber-attacks, and transmission network abnormalities. If these faults are not detected and dealt with promptly, it can severely impact the operation of the whole sensor network. Usually, the IIoT gateway includes multiple network cards with different purposes, such as the Eth0 sensor data collection network card, the Eth1 system maintenance network card, the PPP0 sensor data transmission network card, and the VPN0 virtual network card [4]. By analyzing the flow subsequences of these network cards, relevant information can be acquired to evaluate the performance and stability of the IIoT.

Traditional fault diagnosis methods often focus on the anomalies of individual sensors, which often leads to misjudgments caused by occasional sensor issues. However, factory managers often pay more attention to the occurrence of continuous anomalies, so this article will treat several consecutive anomaly points as one fault. In additionally, for different IIoT applications, it is difficult to label data points due to the varying number of sensors and remote I/O. Based on the above analysis, this article proposes an IIoT fault detection algorithm based on the flow of the sensor network. It consists of the Sequence State Generator based on Unsupervised Learning (SSGBUL) module and the k-Nearest Neighbor based on Indicator Distance (IKNN) classification module. The main contributions of this work are as follows:

1. We design a code generator model for the flow of the sensor network called SSGBUL, and implement it within the subsequence calibration function to reduce the increase in error during the coding process.
2. We convert the multi-dimensional flow sequences of the sensor network to one code sequence representing the operational status of the IioT gateway and identify the fault type via the code sequence using IKNN.

2. Related Work

At present, subsequences can be classified into four categories: (1) distance-based, (2) interval-based, (3) dictionary pattern-based, and (4) neural network-based subsequences.

In terms of distance-based classification, Bagnall [5] proposed an algorithm called Dynamic Time Warping (DTW) [6] that adopts the k-Nearest Neighbor (KNN) classifier for sequence classification. DTW requires defining a large number of subsequence models for pattern matching, which always results in increased time complexity.

In terms of interval-based classification, Deng H, Lines J, and Middlehurst M proposed the Time Series Forest (TSF), Random Interval Spectrum Ensemble [7], and Typical Interval Forest [8] algorithms, which utilize statistical features such as the mean, variance, and slope of the subsequences for matching and use random forest models for classification. Due to the large fluctuations in the numerical characteristics of network flow subsequences, it is difficult to adapt the fixed patterns of statistical features to all types of subsequences in the case of a fault.

In terms of dictionary pattern-based classification, Lin J, Radford A, and He K respectively proposed a pattern packet algorithm [9], a symbol aggregation approximation algorithm [10], and a time series classification based on a word extraction algorithm [11]; these algorithms convert time series data into pattern packets and distinguish between different subsequence categories based on the relative frequency of a pattern packet’s appearance. Because the data span of the network flow is enormous, a set quantity of patterns for one interval value needs to be defined. Otherwise, it will lead to a pattern knowledge explosion.

In terms of neural network-based classification, Wang [12] and Fatwas [13] validated the performance of Convolutional Neural Networks (CNNs) and Residual Neural Networks in classification tasks [14]. In IIoT scenarios, different gateways hold different network flow features, which causes problems for sequence annotation.

3. SSGBUL-IKNN Algorithm

The SSGBUL-IKNN fault detection algorithm consists of the SSGBUL encoding module and the IKNN classification module. There are three submodules in the SSGBUL encoding module: (1) build the model network flow prediction based on unsupervised learning (NFPBUL); (2) encode the network flow sequence using the unified coding module (UCM); and (3) calibrate the input subsequence (CIS). In the classification module, the coding sequences of different network cards are integrated first, and then the fault type is calculated by the IKNN algorithm based on the indicated distance. Figure 1 shows the diagram of the overall network architecture.

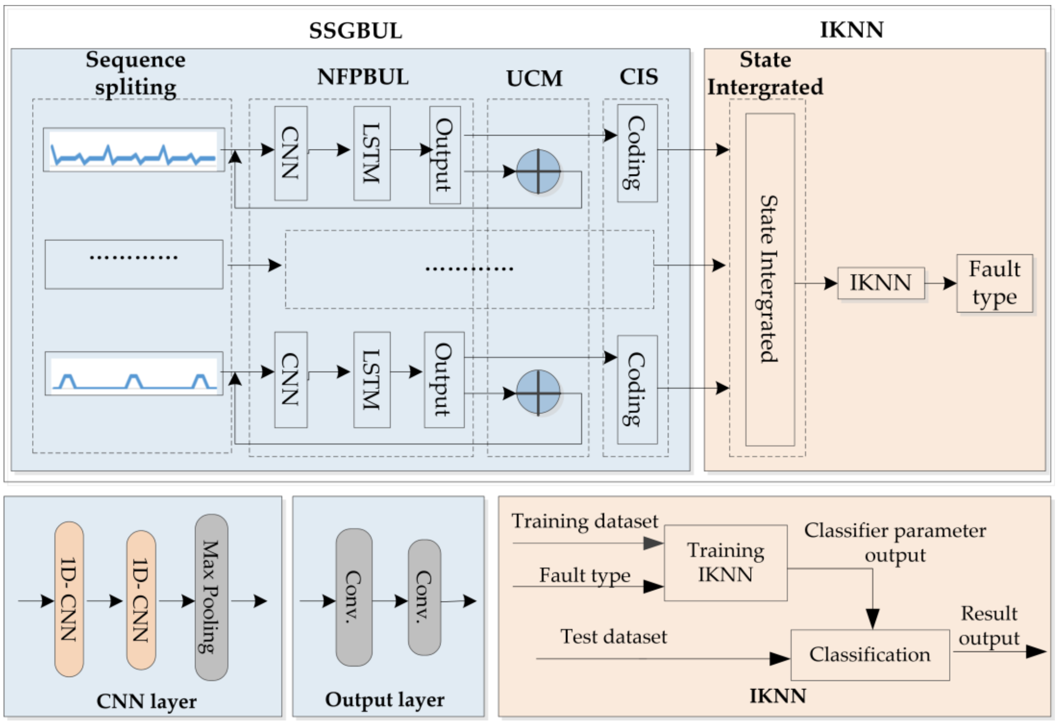


Figure 1. Overall network architecture of SSGBUL-IKNN.

3.1. SSGBUL Model

3.1.1. NFPBUL Prediction Model

For a fixed network card of the IIoT gateway, the network flow is sampled according to a specified time cycle according to Equation (1):

$$D = [d_1, d_2, \ldots, d_m] \tag{1}$$

where m is the collection time. The network flow sequence D is divided into a set of flow subsequences C_d with a length of l using sliding window technology, according to Equation (2).

$$C_d = \begin{Bmatrix} [d_1 & d_2 & \ldots & d_l\ ] \\ [d_2 & d_3 & \ldots & d_{l+1}] \\ & \ldots & & \\ [d_{m-l} & d_{m-l+1} & \ldots & d_m] \end{Bmatrix} \tag{2}$$

The label set C_l for the flow subsequence is constructed according to Equation (3).

$$C_l = [d_{l+1}, d_{l+2}, \ldots, d_m]^{\mathrm{T}} \tag{3}$$

The NFPBUL module consists of four parts: (1) the input layer, (2) the CNN layer, (3) the LSTM layer, and (4) the output layer. The input layer contains a set of network flow subsequences and a set of labels. The CNN layer has two parts: double one-dimensional convolutional components (1D-CNN) [15] and one max pooling component. The first convolutional component extracts the features from the input flow subsequence. The second convolutional component performs the extraction again to obtain an enlarged feature. The max pooling component simplifies the feature and uses it as an input to the decoder. These extracted features will be passed on to the LSTM layers to capture the long-term dependencies of the network flow. The LSTM layer consists of a single LSTM [16] component. The periodicity and regularity of the data are extracted through the LSTM [17] layer. The output layer contains two fully connected components. The first fully connected component is used to enhance the nonlinear ability of the LSTM model, and the second fully connected component is used to output the set of predicted values. Figure 2 shows the network structure of the NFPBUL.

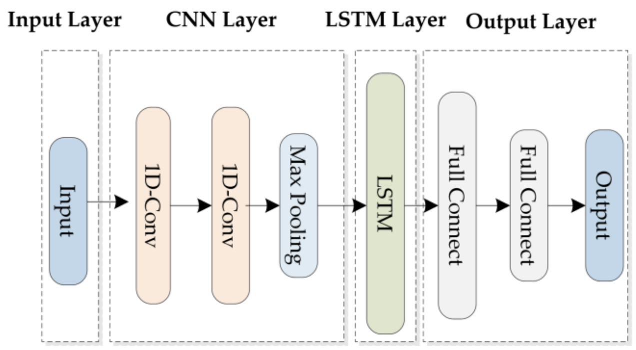


Figure 2. NFPBUL network structure diagram.

The NFPBUL model is trained by the subsequence set C_d and label set C_l [18]. The calculation formula for network flow prediction is shown as follows:

$$C_p = \mathrm{N}(C_d, C_l) \tag{4}$$

where the function N is the NFPBUL prediction model and C_p is the prediction result set.

3.1.2 Coding Model

According to Equation (2), the input sequence is constructed by using a sliding window on the test dataset according to Equation (5).

$$\mathrm{P_t} = \{[d_{t-1-l}, d_{t-l}, \ldots, d_{t-1}]\} \tag{5}$$

The future network flow can be predicted by the NFPBUL network according to Equations (6) and (7).

$$C_p = N(P_t) \tag{6}$$

$$p_t = C_p[0] \tag{7}$$

The UCM module encodes the network flow with values of 1, 0, and -1 [19]. When the difference between the prediction value and the actual value is within the threshold ε, the value of the network flow is set to 0. When the difference exceeds the threshold ε, it means that several system faults have occurred, such as illegal access, cyber-attacks, etc., and the value of the network flow is set to 1. When the difference is below the threshold ε, several faults are predicted to occur, such as sensor disconnection, remote I/O module going offline, etc., and of the value of the network flow is set to -1. The calculation function of encoding is expressed in Equation (8):

$$\mathrm{F_{code}}(d_t, p_t, \varepsilon) = \begin{cases} 1, d_t \in [p_t + \varepsilon, +\infty] \\ 0, d_t \in [p_t - \varepsilon, p_t + \varepsilon] \\ -1, d_t \in [0, p_t - \varepsilon] \end{cases} \tag{8}$$

where d_t is the actual network flow at time t and p_t is the prediction value at time t. The network flow coding sequence can be generated by multiple steps of predicting and encoding according to Equation (9).

$$S_t = [s_t, s_{t+1}, \ldots, s_{t+m}] \tag{9}$$

3.1.3. Subsequence Calibration

During the process of predicting and encoding the flow of the sensor network, if there are no abnormal data, the code of the flow of the sensor network flow can be generated correctly. Otherwise, increases in error will occur, resulting in odd coding. To resolve this issue, we propose a subsequence calibration method to adjust anomalous data in a subsequence based on the difference between the actual network flow and the predicted flow based on the threshold ε. The steps of this method are listed as follows:

(1) Calculate the fixed position flow threshold value in a single data cycle executed on the training dataset using Equation (10). Firstly, calculate the maximum network flow at each selected position. Then, subtract the average network flow to calculate the error value ε_Δ.

$$\varepsilon_\Delta = \max(d_{1*l}, d_{2*l}, ..., d_{n*l}) - \frac{\sum_{i=0}^{n} d_{i*l}}{n} \quad (10)$$

where d_{i*l} is the flow value at a fixed position within the data cycle.

(2) Select the maximum threshold value as the whole sequence threshold according to Equation (11):

$$\varepsilon = \max(\varepsilon_\Delta^1, \varepsilon_\Delta^2, ..., \varepsilon_\Delta^l) \quad (11)$$

where ε_Δ^l is the threshold at a fixed position within the data cycle.

(3) Calibrate the network flow. Based on the difference between the actual and predicted values with the threshold ε, dynamically adjust the sequence item according to Equation (12). If the absolute difference value is greater than the threshold ε, this indicates that the actual value is abnormal, and the network flow subsequence should be constructed using the predicted value. Otherwise, the network flow subsequence can be built using the real value.

$$a_t = \begin{cases} p_t, | d_t - p_t | > \varepsilon \\ d_t, | d_t - p_t | \le \varepsilon \end{cases} \quad (12)$$

where a_t is the reconstructed network flow value at time t.

(4) Obtain the prediction value using the NFPBTSN network model [20] through a newly constructed network flow subsequence [21] based on the calibration function according to Equation (13).

$$p_{t+1} = C_p[0] = N(A_t) = N([a_{t-l}, a_{t-l+1}, ..., a_t]) \quad (13)$$

(5) Generate the network flow code sequence according to Equation (8).

(6) Repeat steps 3 to 5 to generate the final code sequence after multiple rounds of prediction and encoding.

3.2. Classification Algorithm

3.2.1. Integrated Module

For different types of sensor flow, their digital characteristics often make it difficult to accurately describe their trends, while encoded features can effectively express their characteristics based on their contextual features, especially whether anomalies occur. We formed multiple network card flow to one-dimensional encoding features can more effectively describe the operation of IIoT. The code sequences of the flow for sensor network [22] can be integrated into one code sequence, indicating the faults covering the entire gateway [23] according to Equations (14) and (15).

$$S_{all} = \mathrm{Con}([S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8]) \quad (14)$$

$$S_i = \mathrm{Con}([s_1, s_2, ..., s_l]) \quad (15)$$

In Equation (14), S1, S2, S3, S4, S5, S6, S7, and S8 represent the coding sets of the received and sent sequences of each network card. Equation (15) is the state code of a certain point, s_l is the state code of a certain point, l is the sliding window length, and Con is a connection function. Figure 3 shows the integrated diagram [24] of the received and sent code sequences for each network card, such as Eth0, Eth1, PPP0, and VPN0.

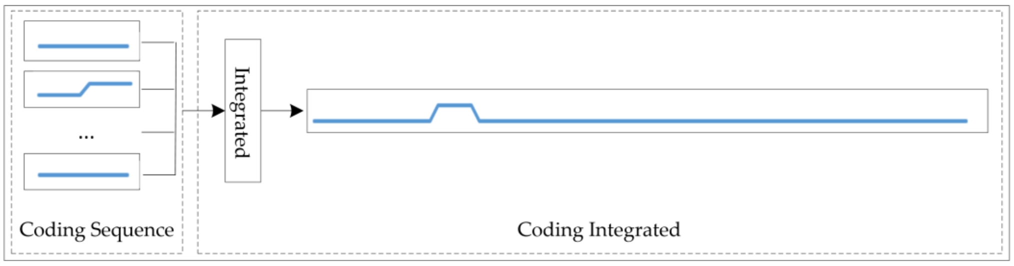
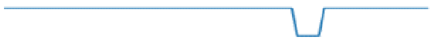
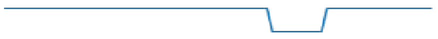
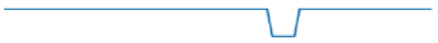
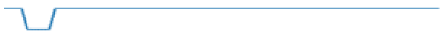
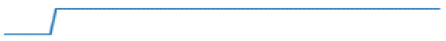
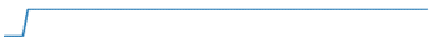
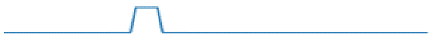
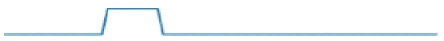
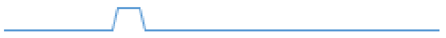
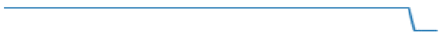
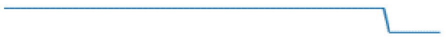
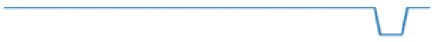
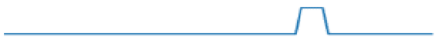
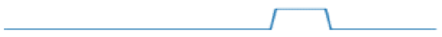
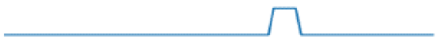
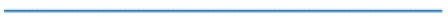


Figure 3. State integration diagram.

3.2.2. Subsequence Type Table

Due to the varying quantity of sensors and remote I/O units connected to the IIoT gateway, it is difficult to define fault sequences according to the network flow for each gateway. Thus, we propose a new way to determine fault sequences using the network flow code to adapt to different gateways [25] in Table 1.

Table 1. Subsequence table.

NO.	Type	Trend Diagram
1	Sensor disconnected, start	
2	Sensor disconnected	
3	Sensor disconnected, end	
4	Remote I/O fault, start	
5	Remote I/O fault	
6	Remote I/O fault, end	
7	Illegal system access, start	
8	Illegal system access	
9	Illegal system access, end	
10	Transmission network abnormality, start	
11	Transmission network abnormality	
12	Transmission network abnormality, end	
13	Cyber-attacks, start	
14	Cyber-attacks	
15	Cyber-attacks, end	
16	Normal	

According to the subsequence trend, the integrated code sequence is divided into two types: linear subsequences and nonlinear subsequences. A linear subsequence means that the state values of each part remain relatively stable without any fluctuations. Nonlinear subsequences refer to sequences in which the state values of various parts in an integrated code sequence undergo trend changes, such as when anomalies begin or end. Table 1 shows the subsequence definition for faults based on the integrated code sequence.

3.2.3. IKNN Classification Algorithm

There are two steps for the KNN algorithm to classify unknown samples: (1) Select the k known classification samples closest to the unknown samples to use as neighbors. (2) Classify the unknown samples based on their class labels or numerical outputs. Based on the indicator function, we propose an improved IKNN algorithm to calculate the distance between sequences [26]. The training set based on Table 1 for the KNN algorithm is constructed according to Equation (16):

$$T = \{(S_1, c_1), (S_2, c_2), \ldots, (S_n, c_n)\} \tag{16}$$

where S_n is the coding sequence defined in Table 1 and c_n is the category number. Traditional distance calculation methods, such as Euclidean distance and DTW, always
face a poor time performance problem when dealing with large classification samples and high-dimensional data. So, a new distance calculation method is proposed in this article based on an indicator function according to Equation (17).

$$\mathrm{I}(\mathrm{m}, n) = \begin{cases} 1, m = n \\ 0, m! = n \end{cases} \quad (17)$$

The distance between the target sequence (TS) and the source sequence (S) in Table 1 is calculated according to Equation (18):

$$\mathrm{D}(TS, \mathrm{S}) =\sim \mathrm{I}(TS_1, S_1) + \ldots + \sim \mathrm{I}(TS_{m*l}, S_{m*l}) \quad (18)$$

where TS_{m*l} is the code value of the sample code sequence, S_{m*l} is the code value of the network flow sequence in Table 1, and m is the dimension of the network flow sequence. Based on the given distance metric, the k network flow subsequences closest to the nearest neighbor are identified according to Equation (19):

$$\mathrm{N_k}(CS) = [y_1, y_2, \ldots, y_k] \quad (19)$$

where y_k represents the category of traffic subsequences. Based on the majority decision, the final category of the code sequence is determined by the most common category in the neighborhood set according to Equation (20):

$$c = \arg\max_{c_j} \sum_{y_i \in \mathrm{N_k}(CS)=1} I(y_i = c_j) \quad (20)$$

where y_i denotes the category of k nearest subsequences and c_j denotes the category defined in Table 1.

4. Data Acquisition

The dataset used in this article contains data sourced from a sewage treatment plant in 2023, which includes data on three industrial IIoT gateways. The factory utilizes the IIoT to achieve automatic operation of the production site. Utilize various types of sensors to monitor the operation of different production areas, and perform on-site automatic adjustment through equipment such as water pumps and fans.

4.1. IIoT Architecture

As Figure 4 shows, the IIoT platform [27] consists of four parts: sensors, a remote I/O unit [28], a gateway, and a cloud server. Sensors are the bridges between the device layer and the gateway and include devices such as thermometers, flow meters, water level meters, PH concentration meters, frequency converters, and so on. The remote I/O unit collects sensor data from various industrial sites and then provides these data to subordinate stations [29] based on different industrial control protocols [30–32] for data sampling by the IIoT gateway. The IIoT gateway is responsible for data sampling and system maintenance. The purpose of a cloud server is to analyze and display the data collected by the gateways.

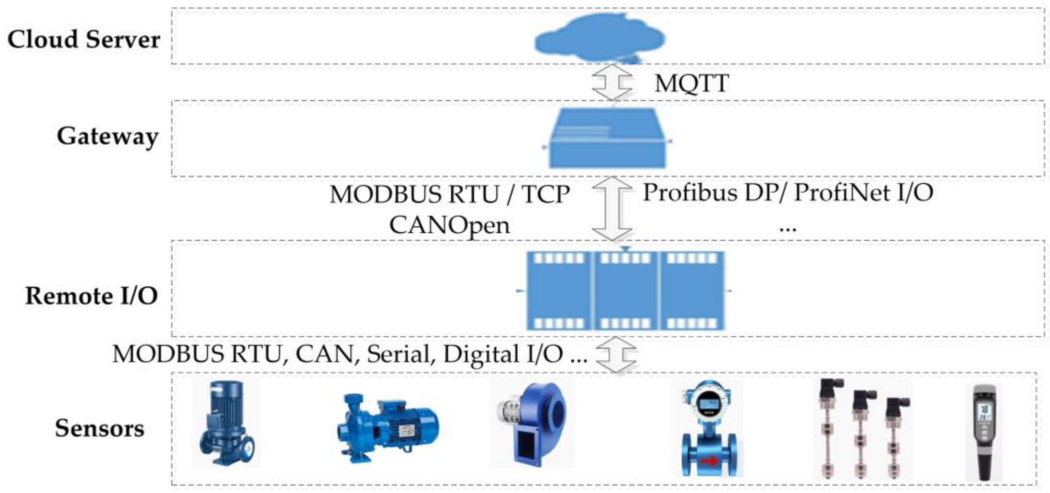


Figure 4. IIoT platform architecture diagram.

4.2. Network Flow Collection Model

Four network cards serving different purposes exist in the IIoT gateway: Eth0, Eth1, PPP0, and VPN0. Eth0 is used to collect sensor information from the remote I/O unit. Before collecting sensor data, other industrial control protocols are converted to the Modbus TCP [33], providing a unified interface for data sampling service. After data collection is complete, the sensor data are transformed to the MQTT[34] protocol format and sent to the cloud server through the PPP0 network card. System administrators can view system data through the VPN0 and Eth1 network cards. Figure 4 shows a diagram of the network cards' functions in the IIoT gateway.

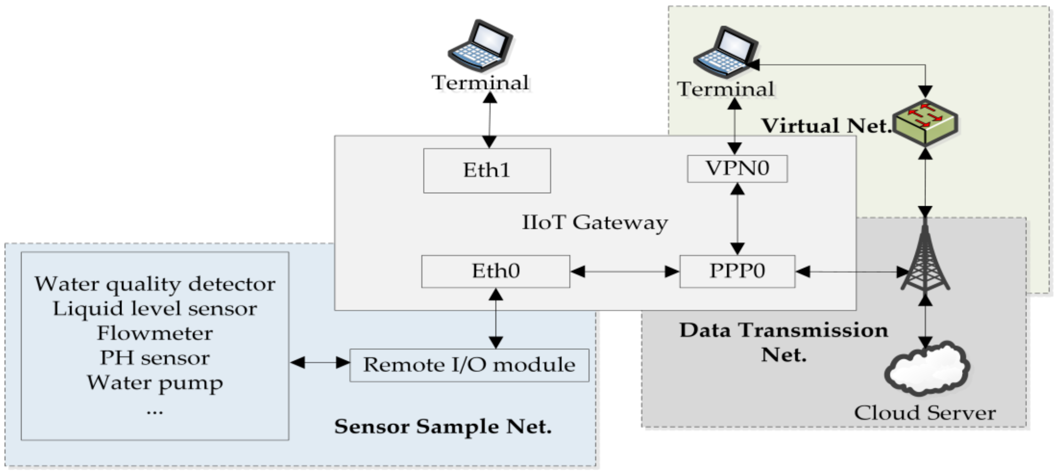


Figure 4. Function diagram for IIoT gateway network card.

4.3. Dataset Introduction

The network flow of each gateway is collected every 10 seconds. Each dataset contains about 10000 records. Dataset 1 sourced from the sewage treatment workshop. When the sewage reaches a certain depth, start the water pump for sewage treatment. Dataset 2 sourced from the automatic dosing workshop, where a PH sensor is used to measure the PH value of the reagent, and the PH value of the reagent is adjusted by dynamically controlling the water pump. Dataset 3 sourced from the production workshop, When the concentration of toxic gases reaches a certain value, the fan device is activated. The specific dataset information is shown in Table 2.

Table 2. Dataset.

Dataset	Remote I/O Amount	Sensor Amount	Sensor Type
1	5	100	water level meter, frequency converter, water pump
2	3	55	PH concentration meter, flow meter, frequency converter, water pump
3	3	35	CO meter, CO^2 meter, blower fan

5. Experiments and Results

The normal function of sensors is the essential to the operation of the IIoT, so it is very important to correctly identify all the issues facing sensors. Based on the above analysis, we conduct five experiments based on three datasets from a sewage treatment plan: a typical abnormal sequence experiment, an ablation experiment, a linear fault detection experiment, a non-linear fault detection experiment, and an accuracy experiment.

5.1. Experimental Metric

In this paper, we estimate the algorithm's accuracy by comparing the annotate and predicted sensor network flow sequences according to Equation (21). The annotate flow sequence is obtained by annotating the original subsequence of the sensor network, and the predicted flow sequence is obtained using the SSGBUL-IKNN algorithm.

$$P = \frac{1}{n}\sum_{k=0}^{n} \mathrm{I}(\mathrm{Raw}(D_t), \mathrm{Detail\_Classfication}(D_t)) \tag{21}$$

where D_t is the original data subsequence, function I is the indicator function, Raw is the function that obtains the annotation of the subsequence, and Detail_Classification is a function specific to the subsequence classification method.

5.2. Typical Abnormal Sequence

Various types of failures can occur in the IIoT [35]. Several typical abnormal sequences [36] are listed below:

1. Sensor disconnection. Sensor data is often sent to cloud servers in MQTT format. The format of MQTT includes data names and data values. The value of data is obtained by converting different types of values into character types, such as long, double, int and so on. When this fault happens, the sensor data will become 0. Moreover, only one digit that is shorter than a regular numeric value will be included, so the length of the converted MQTT transmission packet will be smaller than usual. Figure 5 shows the network flow diagram of sensor disconnection.

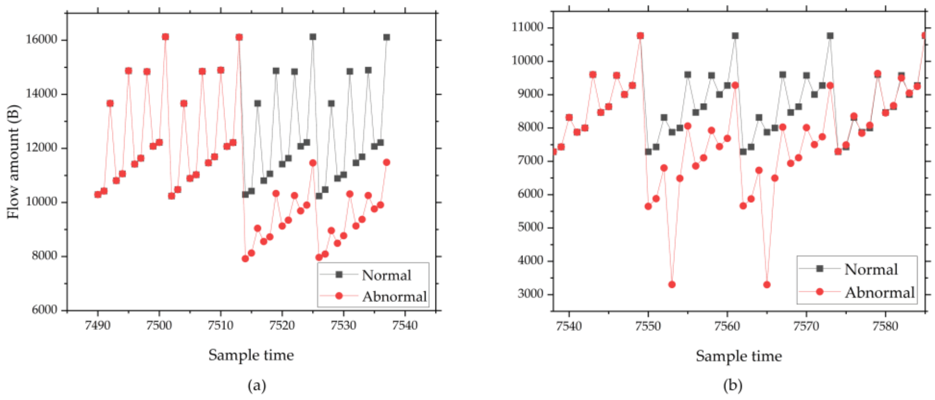


Figure 5. Network flow diagram for sensor disconnection. .

2. Remote I/O offline. When this fault occurs, the IIoT gateway will not collect sensor information connected to this remote I/O unit, as the network is unreachable, so the network flow received by the Eth0 should be decreased. Figure 6 shows the network flow diagram of the remote I/O offline fault.

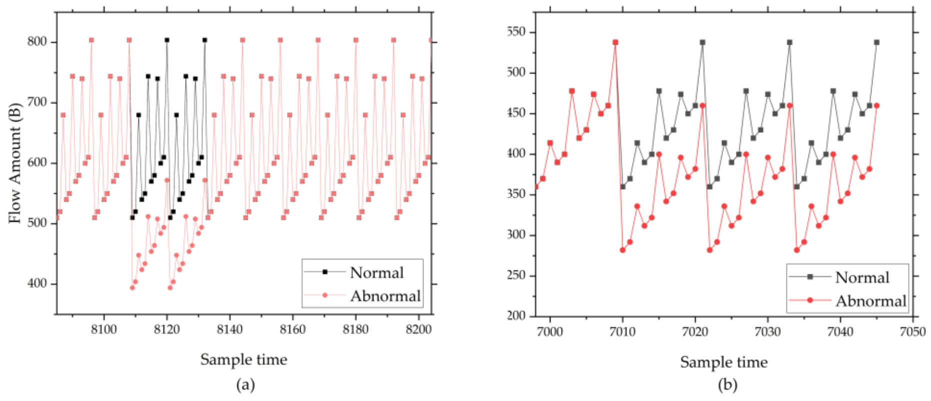


Figure 6. Network flow diagram for the remote I/O offline fault. .

3. Illegal access. Analyzing the received and sent network flow of Eth1, administrators can detect whether the system is being illegally accessed or not. If the network flow increases quickly, it means illegal access is occurring. Figure 7 shows the network flow diagram of illegal access.

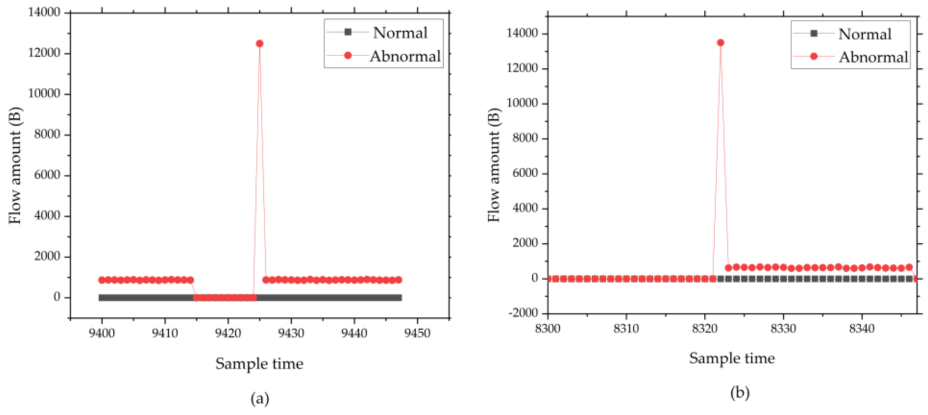


Figure 7. Network flow diagram of illegal access faults. .

5.3. Ablation Experiment

Figure 8 (a) shows part of the flow of the sensor network for the Eth0 card in IIoT Dataset 1. During T1 and T2, one remote I/O unit disconnected, so the sensor data flow related to that unit was reduced. Figure 8 (b.1) shows the coding result of the NFPBUL-UCM model. It can be observed that, during T1 and T2, the abnormal network flow is encoded as a low threshold outlier with a value of -1. During T2 and T3, the abnormal network flow is encoded as a regular code value of 0. During T3 and T4, the abnormal network flow is encoded as a high threshold outlier of 1. Figure 8 (b.2) shows the encoding result of the SSGBUL model. We can see that all abnormal network flows during T1 and T2 are encoded as low threshold outliers (-1), and the regular data are correctly encoded as 0.

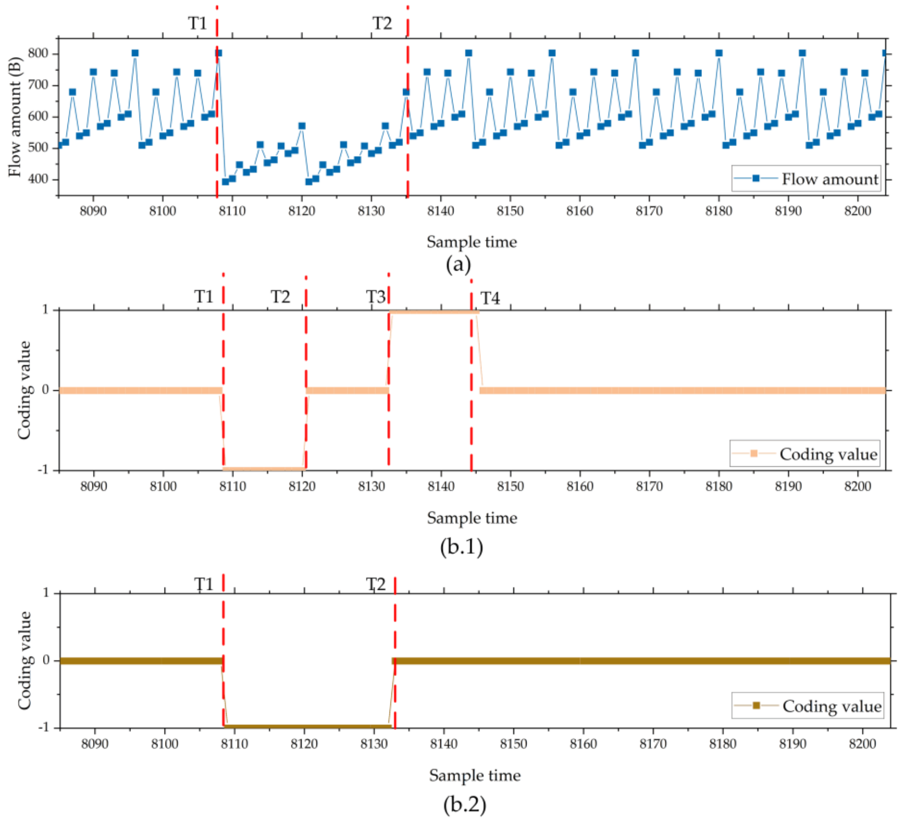


Figure 8. Encoding result diagram.

To sum up, the NFPBUL-UCM encoding module has an incorrect coding problem, for example, the span from T2 to T3 and the span from T3 to T4 in Figure 4 (b.1). The SSGBUL encoding module adopts a subsequence calibration function to adjust the network flow subsequence in time. This contributes to controlling the increase in error observed during multi-step prediction. Therefore, the SSGBUL-IKNN algorithm achieves a better encoding performance.

5.4. Compare Experimental Results

In the process of fault diagnosis, subsequences of different lengths of subsequences have a certain impact on fault diagnosis. In the experiment of linear subsequence classification and nonlinear subsequence classification, we set the subsequence length to 10. In the accuracy experiment, we compared the effectiveness of fault diagnosis with subsequence lengths of 5, 10, and 20 respectively. In this article, the length of the integrated sequence is eight times the length of the subsequence. Below experiments are all based on integrated sequences.

5.4.1. Linear Subsequences Classification

Figure 9 shows the fault detection accuracy of the SSGBUL-IKNN algorithm for linear subsequences, such as sensor disconnection, offline remote I/O modules, illegal system access, cyber-attacks, and transmission network anomalies, as shown below.

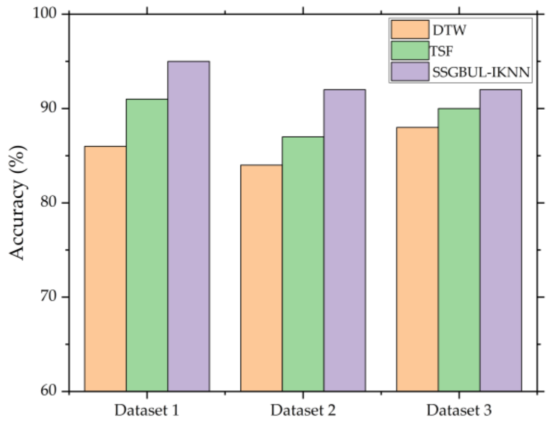


Figure 9. Linear fault detection classification accuracy.

From Figure 9, we can see that the linear fault detection accuracy of SSGBUL-IKNN is significantly higher than the accuracy of DTW and TSF.

Due to DTW's inability to define enough subsequences for matching, the accuracy of identifying abnormal network flow subsequences decreases. Moreover, as the mathematical features of the network flow sequence cannot be efficiently extracted by TSF, the classification accuracy is reduced. Conversely, SSGBUL-IKNN only needs to define the fault sequence based on the coding sequence of 1, 0, and -1, thus narrowing the scope of subsequence definition and improving the fault detection accuracy.

5.4.2. Nonlinear Subsequences Classification

Identifying nonlinear faults can detect the anomalous trends in the IIoT. Figure 10 shows the accuracy of SSGBUL-IKNN in classifying nonlinear fault detection.

As shown in Figure 10, the nonlinear subsequence classification accuracy for SSGBUL-IKNN is significantly higher than that for DTW and TSF.

When a cyber-attack or instance of illegal access occurs, the network flows of Eth1 and PPP0 change significantly, which is a big challenge for feature extraction. For DTW, the considerable change in sequence always occurs outside of the matching subsequence models, which results in misclassifications. For TSF, the statistical features of steeply changing subsequences are always beyond the boundaries of the original definition, which leads to decreases in accuracy. For SSGBUL-IKNN, the numerical sequences of network flow have been converted to code sequences before subsequence classification, promoting fault detection precision.

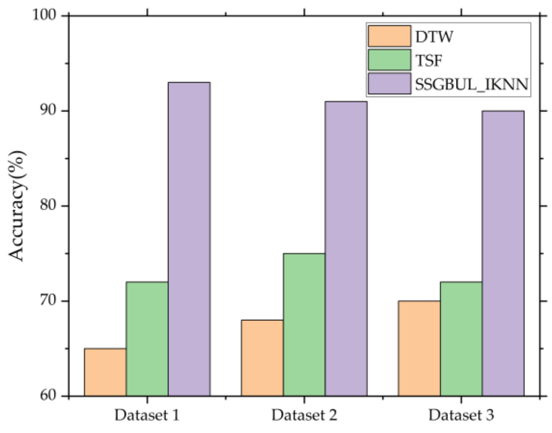


Figure 10. Nonlinear fault detection classification accuracy.

5.4.3. Accuracy Experimental Results

Table 3 shows the fault detection accuracies for the DTW, TSF, and SSGBUL-IKNN algorithms for the three IIoT datasets.

Table 3. Comparison table of fault detection accuracy with subsequence length 5(%).

Subsequence length	Dataset	DTW	TSF	SSGBUL-IKNN
5	Dataset1	89.70	92.27	96.24
	Dataset 2	85.53	88.01	93.43
	Dataset 3	89.81	91.36	93.59
10	Dataset 1	88.60	91.17	95.22
	Dataset 2	84.53	87.01	92.32
	Dataset 3	88.88	90.35	92.48
20	Dataset 1	82.58	89.16	93.21
	Dataset 2	82.34	85.09	90.22
	Dataset 3	86.27	88.24	90.47

According to Table 3, we can see that the accuracy of the SSGBUL-IKNN algorithm reaches 96.24%, 95.22%, 93.21% on IIoT Dataset 1, 93.43%, 92.32%, 90.22% on IIoT Dataset 2, and 93.59%, 92.48%, 90.47% on IIoT Dataset 3 when subsequence length is equal to 5, 10 and 20 respectively; these results are higher than those achieved with the other two algorithms.

The SSGBUL-IKNN algorithm effectively transforms the numerical network flow changes into a flattened code sequence. Compared with DTW, the code sequence only includes the values -1, 0, and -1, so the matching model becomes simpler and more efficient, which is advantageous for improving the accuracy of subsequence classification. Compared with TSF, the statistical features of the code sequence of the subsequences are more apparent than the original network flow sequence. This helps the model adapt to the steep flow sequence. Therefore, the SSGBUL-IKNN algorithm achieves the best fault detection results on the three IIoT datasets.

6. Conclusion

This article proposes a fault detection algorithm called SSGBUL-IKNN based on unsupervised learning encoding. It effectively improves the accuracy of fault detection in IIoT datasets. The main results of the research in this paper are listed as follows:

1. We propose a subsequence calibration function to prevent increases in error during network flow coding, which helps the SSGBUL module to improve the coding performance.
2. We identify the detailed fault categories of the coding sequence indicating the operational status of a gateway integrated with multiple code sequences determined by IKNN based on the indicated distance.

The experimental results show that the SSGBUL-IKNN algorithm achieves an accuracy of over 92% on three IIoT datasets from a sewage treatment plant, thus meeting the requirements of IIoT applications. Future work will aim to optimize the SSGBUL model and improve the fault detection performance with tiny flow fluctuation ranges of sensor networks, thereby achieving better fault detection results.

Author Contributions: Conceptualization, L.D.F.; methodology, Z.L. and C.D.F.; validation, L.D.F. and C.D.F.; writing—original draft preparation, L.D.F.; writing—review and editing, Z.L. and C.D.F. All authors have read and agreed to the published version of the manuscript.

Funding: Funding was provided by the National Natural Science Foundation project (51209167; 12002251) and the Shaanxi Provincial Department of Education Service Local Special Plan Project (22JC043).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data are sourced from a sewage treatment plant in 2023 and are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Huo R, Zeng S, Wang Z, et al. A comprehensive survey on blockchain in industrial internet of things: Motivations, research progresses, and future challenges[J]. IEEE Communications Surveys & Tutorials, 2022, 24(1):88-122.
2. Burke T. Industrial Ethernet's value for communications[J]. Control Engineering, 2022, 69(8).
3. R.K, S.KU. Neuro-Fuzzy-Based Frame Pre-Emption Using Time-Sensitive Networking for Industrial Ethernet[J]. Journal of Information & Knowledge Management, 2021, 20(supp01).
4. Juma M, Monem A A, Shaalan K. Hybrid End-to-End VPN Security Approach for Smart IoT Objects[J]. Journal of Network and Computer Applications,2020,158.
5. Bagnall A, Lines J, and Bostrom A, et al. The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances[J]. Data Mining and Knowledge Discovery, 2017, 31(3): 606-660.
6. Deng H, Runger G, and Tuv E, et al. A time series forest for classification and feature extraction[J]. Information Sciences, 2013, 239: 142-153.
7. Lines J, Taylor S, and Bagnall A. Time series classification with HIVE-COTE: The hierarchical vote collective of transformation-based ensembles[J]. ACM Transactions on Knowledge Discovery from Data, 2018, 12(5).
8. Middlehurst M, Large J, and Bagnall A. The canonical interval forest (CIF) classifier for time series classification[C]. //2020 IEEE International Conference on Big Data. IEEE, 2020.
9. Lin J, Khade R, and Li Y. Rotation-invariant similarity in time series using bag-of-patterns representation[J]. Journal of Intelligent Information Systems, 2012, 39(2): 287-315.
10. T Radford A, Narasimhan K, and Salimans T, et al. Improving language understanding by generative pre-training[J]. 2018.
11. He K, Fan H, and Wu Y, et al. Momentum contrast for unsupervised visual representation learning[C]. //Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020: 9729-9738.
12. Wang Z, Yan W, and Oates T. Time series classification from scratch with deep neural networks: A strong baseline[C]. //Proceedings of the 2017 International Joint Conference on Neural Networks. IEEE, 2017: 1578-1585.
13. Fawaz H I, Forestier G, and Weber J, et al. Deep learning for time series classification: A review[J]. Data Mining and Knowledge Discovery, 2019, 33(4): 917-963.
14. Zhi C, Yuyu S, Xing S, et al. A traffic data interpolation method for IoT sensors based on spatio-temporal dependence[J]. Internet of Things,2023,21.

15. D. A B, Yannis S, Barbara V, et al. Intraclass Clustering-Based CNN Approach for Detection of Malignant Melanoma[J]. Sensors,2023,23(2).
16. Shahzad Z, Nadeem A,Saddam H, et al. A Multi Parameter Forecasting for Stock Time Series Data Using LSTM and Deep Learning Model[J]. Mathematics,2023,11(3).
17. SAHARKHIZAN M, AZMOODEH A, DEHGHANTANHA A, et al. An ensemble of deep recurrent neural networks for detecting IoT cyber attacks using network traffic[J]. IEEE Internet of Things Journal, 2020, 7(9): 8852-8859.
18. Xiangwei C, Wenwen Z, Adrian W, et al. Stacked ResNet-LSTM and CORAL model for multi-site air quality prediction[J]. Neural Computing and Applications, 2022,34(16).
19. Alaghbari A K, Lim S H, Saad M H M, et al. Deep Autoencoder-Based Integrated Model for Anomaly Detection and Efficient Feature Extraction in IoT Networks[J]. IoT, 2023, 4(3).
20. Praveen B K, K. H, R. S, et al. Enabling internet of things in road traffic forecasting with deep learning models[J]. Journal of Intelligent & Fuzzy Systems,2022,43(5).
21. Zhao Na, Sun Hong, Li Quanqi et al. Time series prediction model mWDLNet based on wavelet decomposition and its application research [J]. Small scale microcomputer system, 2022, 43 (03): 561-567. DOI: 10.20009/j.cnki.21-1106/TP.2020-0912.
22. Zhao Jing, Li Jun, Long Chun et al. Unsupervised detection method for RoQ covert attacks based on multi-level features [J]. Journal of Communications, 2022, 43 (09): 224-239. Duan Xueyuan, Fu Yu, Wang Kun. Multidimensional time series anomaly detection method based on VAE-WGAN [J]. Journal of Communications, 2022, 43 (03): 1-13.
23. Jingyi R, Huijuan H, Wenyue X. Big data intelligent tourism management platform design based on abnormal behavior identification[J]. Intelligent Systems with Applications,2024,21200312.
24. Huang Junjie, Xu Xinghua, Cui Xiaopeng, et al. A Time Series Symbolic Aggregation Approximation Method for Fusion of Trend Information [J]. Computer Application Research, 2023,40 (01): 86-90. DOI: 10.19734/j.issn.1001-3695.2022.06.0257.
25. Lu Yi, Wang Peng, Wang Wei. Time series semantic mining algorithm based on subsequence similarity [J]. Computer Engineering, 2022, 48 (10): 88-94. DOI: 10.19678/j.issn.1000-3428.0062832.
26. Mahsa H, Abbas M, Reza G. Stacking: A novel data-driven ensemble machine learning strategy for prediction and mapping of Pb-Zn prospectivity in Varcheh district, west Iran[J]. Expert Systems With Applications, 2024,237(PC).
27. Pino S F A ,Ruiz H P ,Mon A , et al. Systematic literature review on mechanisms to measure the technological maturity of the Internet of Things in enterprises[J].Internet of Things,2024,25101082-.
28. Anonymous .I/O Module Enables Remote Data Capture[J].NASA Tech Briefs,2021,45(5):49-49.
29. Banner Engineering Corp.; Patent Issued for Modbus System Having Actual And Virtual Slave Addresses And Slave Sensors (USPTO 10,805,262)[J].Computers Networks Communications,2020,4905-.
30. Jacek S ,AnneLena K ,Rafał C , et al. Industrial Shared Wireless Communication Systems—Use Case of Autonomous Guided Vehicles with Collaborative Robot[J].Sensors,2022,23(1):158-158.
31. Gateway for IoT integration in PROFIBUS HART systems with extended functionalities[J].M2 Presswire,2023,
32. Yun-Hai C ,Dong-Ying Z ,Jun J , et al. Design of PROFINET I/O Real-time Communication System between PLC Based on S7-1200[J].Journal of Physics: Conference Series,2023,2569(1):
33. Tiago M ,Garcia V S O .Enhanced Modbus/TCP Security Protocol: Authentication and Authorization Functions Supported[J].Sensors,2022,22(20):8024-8024.
34. Hojjati H ,Ho K K T ,Armanfard N .Self-supervised anomaly detection in computer vision and beyond: A survey and outlook[J].Neural Networks,2024,172106106-.
35. Anna C ,TaeYoung K ,Kyung C K , et al. IoT data dissemination scheme for reducing delay in multi-broker environments[J].Internet of Things,2024,25101025-.
36. Aurore C ,Théo T ,Eric T , et al.PhylteR: efficient identification of outlier sequences in phylogenomic datasets.[J].Molecular biology and evolution,2023.