Hindawi
Mobile Information Systems
Volume 2022, Article ID 5988164, 11 pages
https://doi.org/10.1155/2022/5988164

Research Article

Machine Learning-Based Time-Series Data Analysis in Edge-Cloud-Assisted Oil Industrial IoT System

Feng Shi,[1,2] **Liping Yan**,[1] **Xiang Zhao**,[1] **and Richard Xian-Ke Gao**[3]

[1]*College of Electronics and Information Engineering, Sichuan University, Chengdu 610065, China*
[2]*Northwest Oil Field Company, Sinopec Corp., Urumqi, Xinjiang 830011, China*
[3]*Institute of High Performance Computing, A∗STAR, Singapore*

Correspondence should be addressed to Liping Yan; liping_yan@scu.edu.cn

Received 1 September 2021; Revised 13 January 2022; Accepted 25 January 2022; Published 17 February 2022

Academic Editor: Claudio Agostino Ardagna

With the rapid development of the Industrial Internet of Things (IIoT) and edge computing techniques, in situ intelligent sensors are continuously generating increasing and vast amounts of time-series data. In many industrial applications, particularly highly distributed systems positioned in remote areas, repeated transmission of large amounts of raw data onto the remote server is not possible. This poses a significant challenge to the timely processing of these data in IIoT. Analyzing and processing all the raw data remotely in the cloud server is impractical and has very low efficiency owing to network latency and the limited cloud computing resources. Failure of detecting abnormal data may result in major production safety problems. Therefore, businesses are moving machine learning capabilities to the edge to enable real-time actions in the field. In this study, we present a machine-learning-based edge-cloud framework to solve this problem. First, robust random cut forest and isolation forest algorithms are employed at the edge gateway to the collected data for the detection of anomalously changing data. Subsequently, these preprocessed time-series data are transmitted to cloud services for data trend prediction and missing data completion using the long short-term memory recurrent neural network method feed in conjunction with the original sequence of historical data combined with the first-order forward difference data. The experimental results show that the machine-learning-based edge-cloud-assisted oil production IIoT system can improve substantially the efficiency and accuracy of time-series data analyses.

1. Introduction

In recent years, the increasing development of wireless communication, sensor networks, and embedded systems has facilitated the widespread application of the Internet of Things (IoT) in industry, thus leading to the industrial IoT (IIoT). The vast amounts of time-series data generated continuously by smart sensors are essential for the real-time monitoring and intelligent analysis or decision-making of production [1, 2]. For example, sudden changes in time-series data suggest anomalies in the actual production line, and the future trend can be predicted through the periodic pattern of the data [3, 4]. Moreover, the packet loss of temporal data, which is common owing to sensor failure or network congestion, can be monitored and replaced. The voluminous time-series data that which are generated continuously pose a significant challenge to IIoT for efficient data processing and analysis. Analyzing and processing all the raw data in the cloud server remotely is impractical and has very low efficiency owing to the limited network latency and cloud computing resources. Detection failure of abnormal data and potential dangers may cause catastrophic production safety problems or loss of property.

Edge computing and cloud computing provide an efficient alternative to tackle the aforementioned problems. Edge computing is a new computing pattern that offloads computation and storage capacity from the cloud to the edge and only interacts with the cloud off the critical path. Edge nodes are close to the sensing objects or data sources and facilitate data access in a time-efficient manner and improve field device management to satisfy the requirements of low latency, massive connections, and anomalous change detection [5]. Cloud computing is considered a promising information technology infrastructure that can organize significant resources to support on-demand services in remote data centers. A typical method used to improve data analysis efficiency is to perform the primary processing of time-series data at the edge while conducting trend predictions and missing data completion in the cloud based on a series of historical data. The edge-cloud-assisted IIoT system provides a continuum of services for intelligent analysis and application of time-series data.

This study presents an edge-cloud-assisted system by integrating edge computing and cloud computing technologies in IIoT for efficient time-series data analysis, including anomalous data detection, time-series data trend prediction, and missing data replacement. The proposed system consists of three layers: a smart perception layer, an edge computing layer, and a cloud computing layer to implement different functions. The main contributions of this study are summarized below.

We propose an edge-cloud-assisted IIoT system based on the Aliyun Link-Edge runtime environment that can link in and connect the registered field devices, collect the time-series data through the field wireless network, and transmit the data to remote data centers via a public base station. The perceived data by sensors can be cached at the edge, and the data can be reuploaded in case of network congestion.

Because of the limited computing and storage resources at the edge, we initially used the robust random cut forest (RRCF) and isolation forest algorithms to detect time-series data anomalies at the edge to determine whether the data are anomalous by anomaly scores. These scores can be used to monitor a particular event-driven change and make alerts for predictive maintenance. We use a long short-term memory (LSTM) recurrent neural network to predict the future trend of the time-series data and complete the missing data caused by device failure or communication congestion from the original two-dimensional sequence of historical data combined with the first-order difference data of the first-dimensional data in the cloud. LSTM can reveal dynamic temporal behavior in a time sequence. The developed LSTM neural network model can obtain accurate predictions of subsequent data trends and exhibit data characteristics.

The remainder of this study is organized as follows. Related work is briefed in the following section. The overall design of the collaborative edge-cloud IIoT system is then developed. The subsequent section describes the time-series data anomaly detection at the edge. Subsequently, the trend prediction based on historical data using the LSTM method and performance evaluations is discussed. Finally, we summarize our work in the conclusion.

2. Related Works

In this section, we review some existing research works relevant to our study. The architecture and functionalities of edge-cloud collaboration have been addressed in several prior studies but in different application scenarios, wherein the specific functions of the edge layer and the cloud layer were implemented differently.

Initially, the difficulties of traditional centralized cloud computation and storage in remote data centers with the emergence of IoT systems were analyzed for the existing edge-end collaboration IIoT system schemes [6]. Moreover, the edge advantages of a series of latest technologies, such as fog computing and mobile edge computing, were compared to provide a reference for possible future research directions and applications. A mobility-driven cloud-fog/edge collaborative framework was proposed [7], which has a device, edge/fog, and cloud layers, and which employs machine learning in the cloud layer to predict the location of moving agents based on spatial-temporal mobility data. A flexible framework has been designed by integrating fog and cloud computing for data processing and storage [3]. Moreover, a novel architecture incorporating a device's trust evaluation mechanism and edge network service template with cloud and edge computing to improve system security was presented [8]. Other related studies have mostly focused on how to balance computing loads at the edge and in the cloud, taking full advantage of their computing strengths and adapting to connected devices and network security [9, 10].

From this retrospective analysis, it is clear that the particular function at different layers needs to be considered for different collaborative edge-cloud application scenarios. IIoT systems must be able to detect changes in the collected data timely at the edge for monitoring device status. Various researchers have explored different machine learning algorithms to detect anomalies and discover hidden patterns in time-series data [11]. Previous research has shown that ECHAD, an embedding-based, one-class learning, and dynamic change detection algorithm, can detect anomalous changes in time-series data generated by smart grids [12]. Several studies have been reported to diagnose sensory anomalies based on convolutional neural networks combined with long- and short-term memory network or median filter stacked long and short-term memory-exponentially weighted moving average [13, 14].

Some studies have been devoted to forecasting future trends and completing missing data based on historical data using deep learning methods. As a big data-driven analytics method, deep learning has exhibited considerable potential in several areas, including the formulation of predictions and the learning of temporal contextual information. However, as deep learning requires high-computation capability and storage capacity to analyze datasets, it is generally deployed in cloud servers [15–17]. Some related work on edge-cloud assisted computing is summarized in Table 1. These existing publications provide an in-depth discussion on different computing tasks in edge and cloud layer from different perspectives and use different computing methods for anomaly detection and future trend prediction.

In this study, a collaborative edge-cloud system with machine learning and deep learning is proposed to tackle the problems mentioned above. First, we concentrate on offloading the computation capabilities to the edge node and executing a machine-learning algorithm to detect anomalous data points, thereby reducing the possibility of network congestion and mitigation of computational overloads in the cloud. Second, we use deep learning to predict future data

Table 1: Summary of the differences of related works.

Aspects	Survey papers	Differences
Edge-cloud computing	[6]	Draw an overall picture of the ongoing research efforts and future research directions on edge and cloud computing
	[7]	A framework of edge, fog, and cloud computing system for spatial-temporal mobility data prediction
	[3]	A flexible framework used to solve data collection, processing, and secure storage by integrating fog computing and cloud computing
	[8]	A novel architecture that incorporates a trust evaluation mechanism and edge-cloud network service to improve system security
	[9, 10, 16]	Balance computing loads at the edge and in the cloud and take full advantage of the computing strengths and network security
	[17–19]	Rethinking deep learning requires a high-computation capability and storage capacity to analyze time-series data
Anomaly data detection	[11]	A comprehensive survey of the background and challenges of anomaly detection techniques, examples of applications for anomaly detection
	[12]	A novel change detection method, leverage embedding techniques, one-class learning, and a dynamic detection approach on real data, thus avoiding false positive detections
	[13, 14]	A new algorithm for recognizing sensory anomalies based on computational neural networks (CNN) combined with long short-term memory (LSTM) or median filter stacked LSTM-exponentially weighted moving average
	[15]	Study collective contextual anomalies based on semisupervised deep learning, time-series modeling, and graph analysis for contextual anomaly prediction
Predicting future trends	[20]	A novel least absolute shrinkage and selection operator and LSTM integrated forecasting model for predictions
	[21]	A new sequence-to-sequence deep learning model and LSTM can be used to utilize both the past and future information for recovering missing data in wireless sensor networks
	This article	A collaborative edge-cloud system with machine learning can be used to tackle anomaly data detection, future data trend prediction, and missing data replacement. Concentrating on offloading the computation capabilities to the edge node, thereby reducing the possibility of network congestion in the cloud

trends in stream data and present a feasible means for completing missing data. Finally, we select a real industry dataset to validate the effectiveness of our methods. As far as we know, this is the first time that RRCF and isolation forest algorithms have been used at the edge to monitor anomalies in a real-time industrial dataset.

3. Edge-Cloud-Assisted IIoT System Design

Edge computing is a new computing paradigm that offloads computing and storage resources from the cloud to the edge, which is close to the data source, enabling sensor access management and data preprocessing at the edge to satisfy the requirements of low latency and massive connectivity. Unlike the traditional cloud-based system, the raw data collected by the devices must be transmitted to the cloud for data processing and analysis. Therefore, exploring a collaborative edge-cloud mode that can both efficiently detect data anomalies at the edge and predict data trends in the cloud is necessary [18, 19, 22–24].

An overview of the three-tier collaborative edge-cloud system with machine learning and deep learning based on the link-edge environment is given in Figure 1. The architecture and its tiers are then described. The edge computing layer is located between the cloud and the smart perception layer and supports field devices access management and collaboration with the cloud. A private communication protocol driver for device access and two machine learning algorithms for time-series data anomaly analysis were deployed at the edge layer. Moreover, the LSTM recurrent neural network algorithm was embedded in the cloud computing layer for future trend prediction.

An edge computing node carries device access and data preprocessing functions. It usually refers to a small device with computing and storage capabilities, which can be an edge gateway or a miniaturized embedded industrial computer. The core function of an edge node is to enable device registration and device management at the edge; it has limited local resources to perform real-time data preprocessing, while publishing data to the cloud for future in-depth analysis. A functional view of edge computing is shown in Figure 2.

4. Time-Series Data Anomaly Detection

IIoT has been extensively used and researched for many applications; this is mainly because the frequency and efficiency of data collection in these devices have increased, thus generating a large amount of time-series data, which can monitor the status of a specific equipment over time. It is often necessary to identify anomalous changes or unusual states within a system being monitored by the sensors. This process is often referred to as an anomalous detection or outlier detection and is designed to discover the value of the changed data, identify defective equipment, spot quality issues, and alert

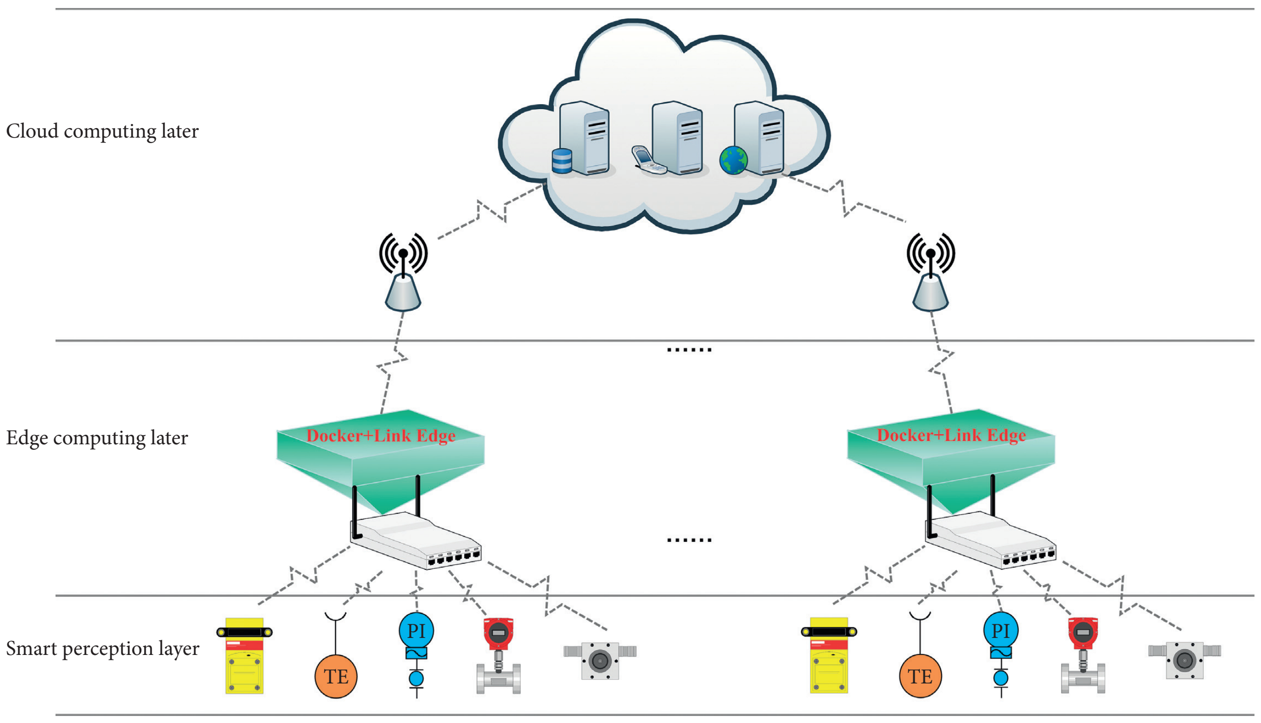

Figure 1: Overview of collaborative edge-cloud Industrial Internet-of-Things (IIoT) system architecture.

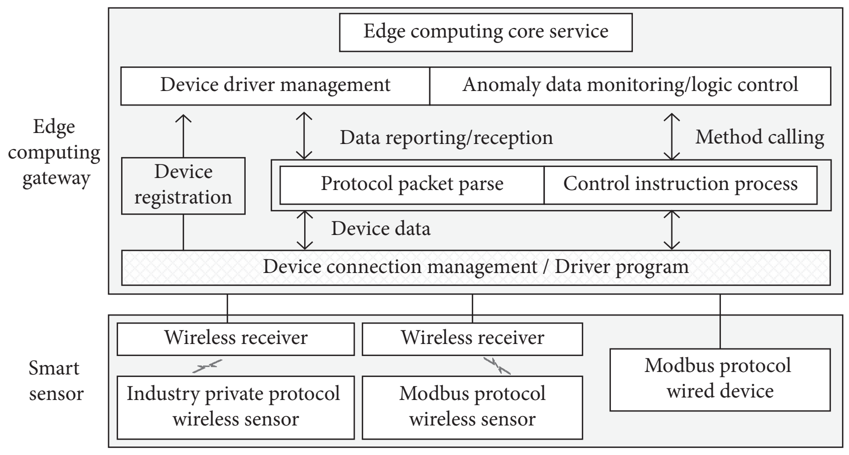
Edge computing core service
Device driver management
Anomaly data monitoring/logic control
Data reporting/reception
Method calling
Edge computing gateway
Device registration
Protocol packet parse
Control instruction process
Device data
Device connection management / Driver program
Wireless receiver
Wireless receiver
Smart sensor
Modbus protocol wired device
Industry private protocol wireless sensor
Modbus protocol wireless sensor

Figure 2: Functional view of edge computing.

supervisors to focus on abnormal changes. Time-series data anomalies can be point, contextual, or pattern anomalies. Because of the massive and real-time nature of time-series data, single-point detection needs to be performed at the edge and detection accuracy needs to be improved to avoid incorrect information alarms. Therefore, we apply two machine learning methods for anomaly data detection at the edge and compare their advantages and disadvantages separately.

4.1. RRCF Method. RRCF is an unsupervised method used for the detection of anomalies in dynamic data streams, which is efficient in processing vast amounts of data streams and suitable for high-dimensional data [11]. It can effectively reduce the effect of unrelated dimensions by removing duplicates and near-duplicates from the time-series data when constructing the forest, thus avoiding hiding the existence of a group abnormal value, and using statistically significant anomaly scoring indicators to assess anomaly values.

By using a sketch to construct summaries of time-series data, constructing an integration of spatially divided binary trees on a set of points, and then calculating anomaly scores based on the effect of insertion or deletion of each point on the remaining data, the RRCF algorithm can effectively detect anomalies on stream data, while adapting to changes in input data and handling collusive outliers. A robust random cut tree (RRCT) on point set S is generated as follows:

(1) Choose a random dimension proportional to $l_i/\sum_j l_j$, where $l_i = \max_{x\in S} x_i - \min_{x\in S} x_i$.

(2) Choose $X_i \sim \text{Uniform}[\min_{x\in S} x_i, \max_{x\in S} x_i]$.

(3) Let $S_l = \{x | x \in S, x_i \leq X_i\}$, and $S_r = S/S_l$; recursive execution on the left tree S_l and right tree S_r, respectively, builds a binary tree.

(4) Calculate anomaly scores based on the changes in model complexity due to the insertion or deletion of each input data point.

The algorithm of constructing RRCT is provided in Algorithm 1.

The RRCF can construct a set of independent random cut trees according to the aforementioned steps. The tree model complexity of the rest of the data caused by the insertion and deletion of data points is assessed to detect anomalous changes. Given a set of points Z and a point $y \in Z$, let $f(y, Z, T)$ be the depth of y in the tree T. The model complexity is denoted as follows:

$$|M(T)| = \sum_{y\in Z} f(y, Z, T). \tag{1}$$

If we were to remove x, the new model complexity is

$$|M(T')| = \sum_{y\in Z-\{x\}} f(y, Z - \{x\}, T'), \tag{2}$$

where $T' = T(Z - \{x\})$ is a tree over set points $Z - \{x\}$. The displacement of a point x is defined as an increase in the model complexity of all other points for a set Z and can be denoted as follows:

$$\text{DISP}(x, Z) = \sum_{T, y\in Z-\{x\}} \Pr[T](f(y, Z, T) - f(y, Z - \{x\}, T')), \tag{3}$$

where $\Pr[T]$ is the probability of a change in the depth of the sibling node of the leaf node containing x in tree T due to insertion or deletion at point x. The expected displacement associated with a point x is simply equal to the number of leaves in the subtree beneath the sibling node of x.

The displacement of a point is easy to find from the aforementioned formula. Moreover, when the anomalous data points are repeated in a small area, the displacement of the anomalies is very small. This phenomenon is known as outlier masking. Duplicates and near-duplicates are natural phenomena; anomaly detection must remove duplicates and near-duplicates. Therefore, we define "collusive" displacement as follows:

$$\text{CODISP}(x, Z, |S|) = E_{S\subseteq Z,T}\left[\max_{x\in C\subseteq S} \frac{1}{|C|} \sum_{y\in S-C} (f(y, S, T) - f(y, S - C, T''))\right], \tag{4}$$

where C is the set of "collusive" points to be removed alongside point x, S is a true subset of Z in a streaming choice of the C set, and $T'' = T(Z - C)$ is a tree over a set points $T(Z - C)$. The outliers correspond to large CODISP values. See [25] where the algorithm is detailed.

4.2. Isolation Forest Method. The isolation forest algorithm randomly samples the dataset and constructs a random binary tree, called an isolation tree (iTree), with either two children per node or a leaf node with no children. Subsequently, the method integrates multiple iTrees into a forest, called an isolated forest (iForest), and transverses the input data through the nodes of each tree to detect whether it is an anomaly according to the average path depth. Each iTree is constructed as follows [26]:

(1) Choose ψ sample points randomly as subsamples from a given dataset and construct an initial iTree.

(2) For a list of attributes in the input dataset, randomly select an attribute q and randomly select a split point p between the max and min values of the attribute q.

(3) Based on the value of p, classify each record; place the record value $< p$ in the left node and record the records with values $\geq p$ value in the right node.

(4) Perform steps 2 and 3 recursively, and construct new tree nodes until there is only one data in the subsampled dataset that can no longer be split, or until the tree height has reached the initially defined limits.

Isolation forest is an ensemble of iTrees, using random partition to iteratively isolate every data point to create each iTree. The algorithm is recommended in Algorithm 2.

For a test data point x, iterate through each iTree and calculate the depth at which x falls on each tree. The length of the path from the leaf node to the root node is recorded as $h(x)$ to determine whether a record x is an anomaly. The average of all $h(x)$ is $E(h(x))$ and gives the average path length of the binary search tree in iForest as $c(\psi)$.

$$c(\psi) = 2H(\psi - 1) - [2(\psi - 1)/\psi], \tag{5}$$

where $H(i)$ is the harmonic number, which can be estimated by $\ln(i) + 0.5772156649$ (Euler's constant). As $c(\psi)$ is the average of $h(x)$ given ψ, we use it to normalize $h(x)$.

We use the anomaly scores $s(x, \psi)$ to assess whether a data point is anomalous. If instances return values of $s(x, \psi)$ very close to unity, then they are anomalies. By contrast, if instances return values that are much smaller than 0.5, they are considered as normal instances.

$$s(x, \psi) = 2^{-E(h(x))/c(\psi)}, \tag{6}$$

where $E(h(x))$ is the expected path length of $h(x)$ from a collection of iTrees. Using the isolation forest and random cut forest algorithm to detect anomalous changes in time-

```
    Precondition: $S$ □ $\mathbb{R}$□, unlabeled set of point in which to find anomalies
(1) function Initializer RRCT($S$)
(2)   $x_i^l \leftarrow \min_{x \in S} x_i$
(3)   $x_i^h \leftarrow \max_{x \in S} x_i$
(4)   $B(S) \leftarrow [x_1^l, x_1^h] \times [x_2^l, x_2^h] \times \cdots \times [x_d^l, x_d^h]$ △ minimal bounding box
(5)   if $|S| = 1$ then
(6)     $c_l \leftarrow$ None △ left child is empty
(7)     $c_r \leftarrow$ None △ right child is empty
(8)     return External node of only $S$
(9)   else
(10)    Choose a dimension, $i$, proportional to $l_i / \sum_j^d l_j$
(11)    Choose a cut $C \sim$ Uniform$[x_i^l, x_i^h]$
(12)    $S_l \leftarrow \{x | x \in S, x_i \leq C\}$ △ left child tree
(13)    $S_r \leftarrow S/S_l$ △ right child tree
(14)    $c_l \leftarrow$ Initialize RRCT($S_l$) △ recurse on left
(15)    $c_r \leftarrow$ Initialize RRCT($S_r$) △ recurse on right
(16)    return Internal node
(17)  end if
(18) end function
```

Algorithm 1: Setting up a robust random cut tree

```
    Precondition: $\psi$ □ $\mathbb{R}$□, unlabeled set of point in which to find anomalies; $H$ is the height limit specified; $h$ is the current depth,
    which is initially equal to zero
(1) function Initializer IT($S$, $h$)
(2)   if $|S| = 1$ or $h = H$ then △ a point has been isolated
(3)     $count \leftarrow |S|$
(4)     $c_l \leftarrow$ None △ left child is empty
(5)     $c_r \leftarrow$ None △ right child is empty
(6)     return External node of only $S$
(7) else
(8)     Choose a dimension, $q \sim$ Uniform$[1, p]$
(9)     $x^l \leftarrow \min_{x \in S} x_q$
(10)    $x^h \leftarrow \max_{x \in S} x_q$
(11)    Choose a cut $C \sim$ Uniform$[x^l, x^h]$
(12)    $S_l \leftarrow \{x | x \in S, x_q \leq C\}$ △ left child tree
(13)    $S_r \leftarrow S/S_l$ △ right child tree
(14)    $c_l \leftarrow$ Initialize IT($S_l$, $h+1$) △ recurse on left
(15)    $c_r \leftarrow$ Initialize IT($S_r$, $h+1$) △ recurse on right
(16)    return Internal node
(17)  end if
(18) end function
```

Algorithm 2: Setting up an isolation tree

series data, the anomaly score was calculated according to the average path and the collusive displacement, respectively. The comparative analysis result of the anomaly scores after normalization is shown in Figure 3.

The time-series data curves of the oil-well pipeline pressure collected by our proposed system are shown in Figure 4. There are approximately nine anomalies in total, which are manually labeled with the green vertical line. A two-dimensional time-series of 5370 data points of pipeline pressure were tested, and an abnormal score was rendered to evaluate the efficiency and accuracy of anomalous change detection with random cut forest and isolation forest algorithms. As shown in Figure 3, the isolation forest algorithm can monitor a higher number of anomalous data values than the random cut forest method; however, these results may lead to an increase in false alarm alerts. The efficiency and sensitivity of the anomaly scores are illustrated in Figure 4. The random cut forest is more robust than the isolation forest algorithm; however, the latter has higher sensitivity and has a response time ahead of the random cut forest method; thus, the combination of the two algorithms can respond to production anomalies better.

To evaluate the time efficiency of the anomaly detection algorithms, we use a slice of the pressure data (approximately 6300 data points) onto an oil-well to test for the edge and the cloud. The comparison results are shown in Table 2.

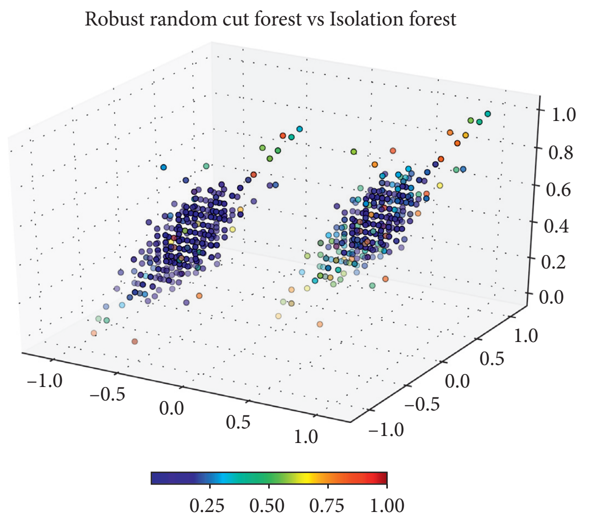

Figure 3: Time-series data rendered by anomaly scores.

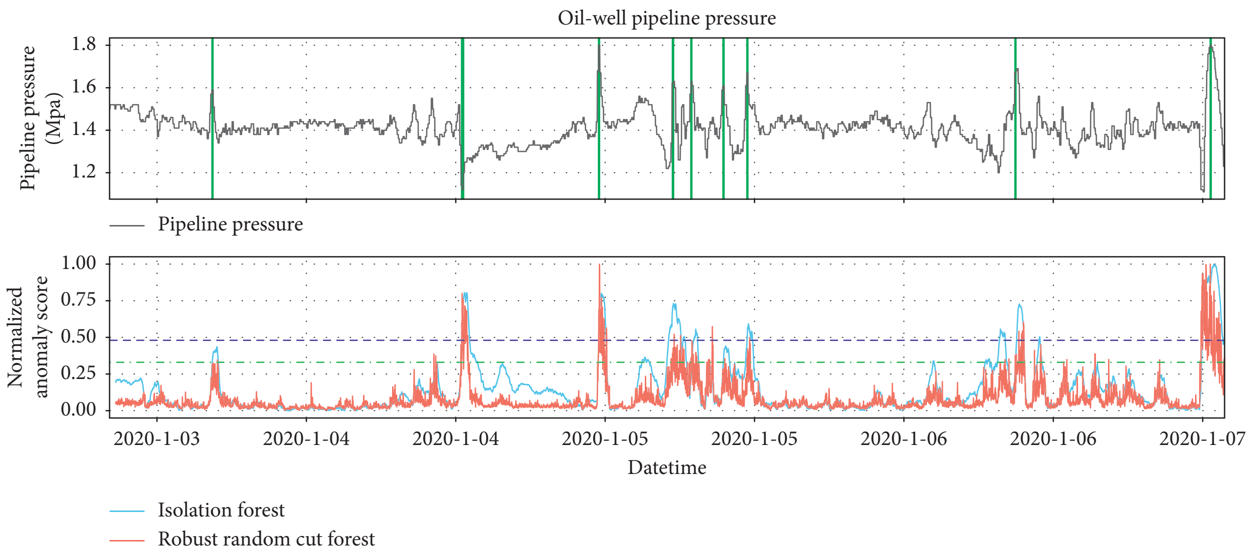

Figure 4: Comparison of the calculated anomaly scores of time-series data.

A MOXA UC-8410A embedded computer has customized a Freescale Cortex-A7 dual-core 1 GHz central processing unit (CPU) and 1 GB synchronous dynamic random access memory to run application software. A HUAWEI RH5885 v4 high-performance server uses an Intel Xeon E7 2.0 GHz CPU and 512 GB dual-inline memory module memory to run the most computationally intensive parts. The average time for detecting a slice of data anomaly at the edge is approximately seconds; thus, the algorithms provide a time-efficient manner for production applications. Uploading only anomaly data and alarm rules for the cloud can also reduce network transmission costs.

Table 2: Anomaly detection runtime comparison.

Edge/cloud	Robust random cut forest (s)	Isolation forest (s)
MOXA UC-8410A	5.26	0.554
HUAWEI RH5885 v4	0.613	0.065

5. Time-Series Data Future Trend Prediction

5.1. Time-Series Data Prediction Using LSTM. Using a series of historical data to predict future trends and complete missing data requires a large amount of computing and storage resources, which have to be inevitably implemented in the cloud. In this section, we use an extensively applied LSTM to perform time-series data analysis, which can both predict trends and fill in missing data [20, 21]. The LSTM introduces a called gate structure for each self-looping cell that mimics the information conduction pattern of biological neurons, storing long-term sequence information without any additional adjustment.

The LSTM architecture consists of a series of neuronal cells, wherein each cell contains three gates, a forget gate f_t, an input gate i_t, an output gate o_t, and one tanh layer, as

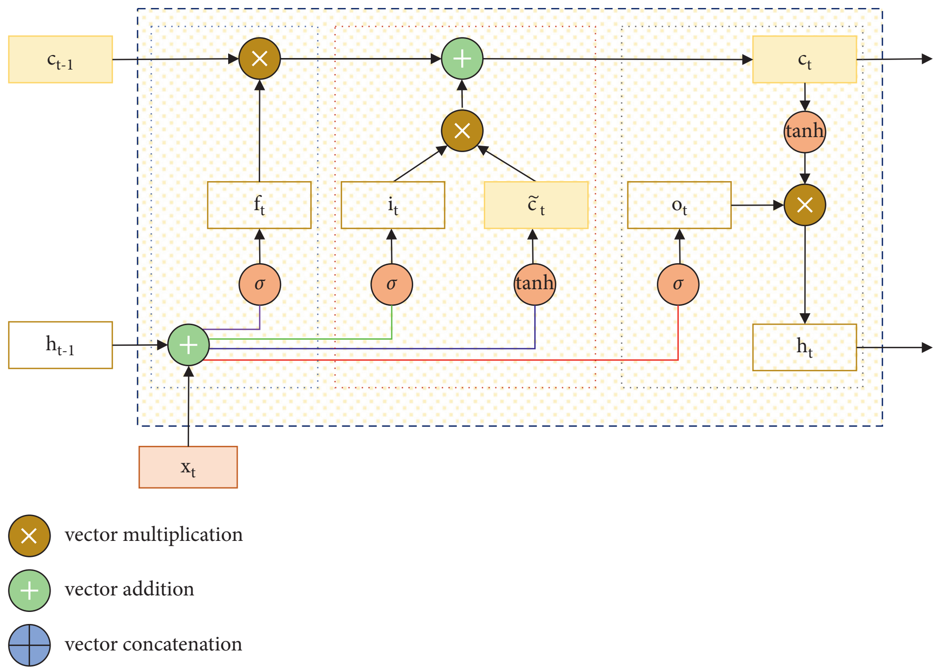

Figure 5: Long short-term memory (LSTM) cell structure.

shown in Figure 5. The neuron cell state is the crucial variable that carries information from the previous step. The gate in the interaction layer can partially remove the state of the previous step and add new information to the current step based on the hidden state of the previous step and the input of the current step.

The first interaction layer, called the forget gate, determines which part of the information from the previous step should be removed and passed down from the cell state. The forget gate is described by

$$f_t = \sigma\left(w_f \cdot [h_{t-1}, x_t] + b_f\right), \tag{7}$$

$$i_t = \sigma\left(w_i \cdot [h_{t-1}, x_t] + b_i\right), \tag{8}$$

$$\tilde{c}_t = \tanh\left(w_c \cdot [h_{t-1}, x_t] + b_c\right), \tag{9}$$

$$c_t = f_t * c_{t-1} + i_t \times \tilde{c}_t, \tag{10}$$

$$o_t = \sigma\left(w_o \cdot [h_{t-1}, x_t] + b_o\right), \tag{11}$$

$$h_t = o_t \times \tanh\left(c_t\right). \tag{12}$$

The output f_t of this layer is between zero and one. The second interaction layer, called the input gate, determines what new information should be added to the cell state. The input gate is described by (8). The output of this layer determines which information is retained and updated. The third interaction layer corresponds to the tanh layer, which creates a new candidate value that can be added to the cell state, as described by (9). The symbol f_t from the forget layer determines the type of information which is forgotten in c_{t-1}, whereas c_{t-1} of the old cell state carrying the memory information is combined with the result of the input value i_t multiplied by the candidate value $\tilde{c}_t$ containing the new information. Subsequently, c_t is obtained, as shown in (10). The final layer is the output gate layer, which generates the output value h_t of the neuron cell by joining the information from current cell states, as shown in (12). These three control gates allow the LSTM network to learn long-term temporal dynamics from the input time-series data, thus predicting future trends and substituting missing data.

5.2. LSTM Model Structure and Training. The proposed scheme using the LSTM recurrent neural network algorithm is implemented using Python 3.6 and TensorFlow 2.0. The LSTM network structure is built with three hidden layers and one dense fully connected layer. The hidden state in each memory layer has 64 neurons and is fully connected to the output layer, yielding single sequential values to predict future data trends. A three-dimensional vector with a batch size of 60 is used as an input to the model and is transmitted to the output dense layer to generate the final results, which represent the predicted future value. To prevent overfitting, a dropout layer is added behind the hidden layers for regularization. After repeated testing, it is found that the accuracy of the training set is the highest when the dropout is equal to 0.20. The adopted LSTM model used to predict the future data trend is shown in Figure 6. It mainly includes data preprocessing, data normalization, data division, prediction model establishment and evaluation, and the predicted results.

The training dataset of the model uses the first length of 50% of a slice of time-series data, approximately 3200 two-dimensional data for model training, the middle 25% of the data for testing, and the latter 25% slice for validation. The

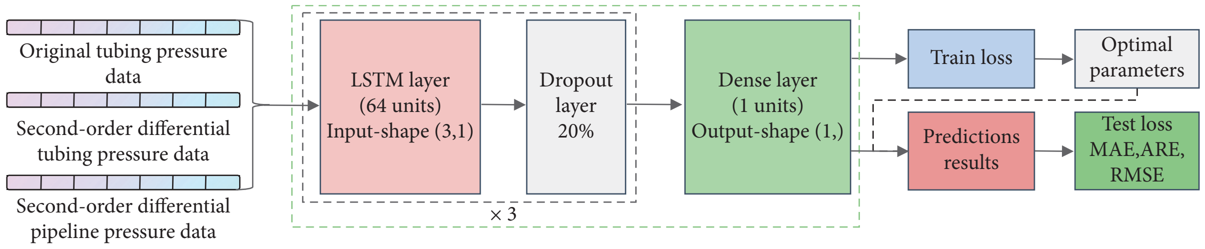
Original tubing pressure data
Second-order differential tubing pressure data
Second-order differential pipeline pressure data
LSTM layer (64 units) Input-shape (3,1)
Dropout layer 20%
× 3
Dense layer (1 units) Output-shape (1,)
Train loss
Optimal parameters
Predictions results
Test loss MAE,ARE, RMSE

Figure 6: Schematic of the LSTM-based pipeline pressure prediction process.

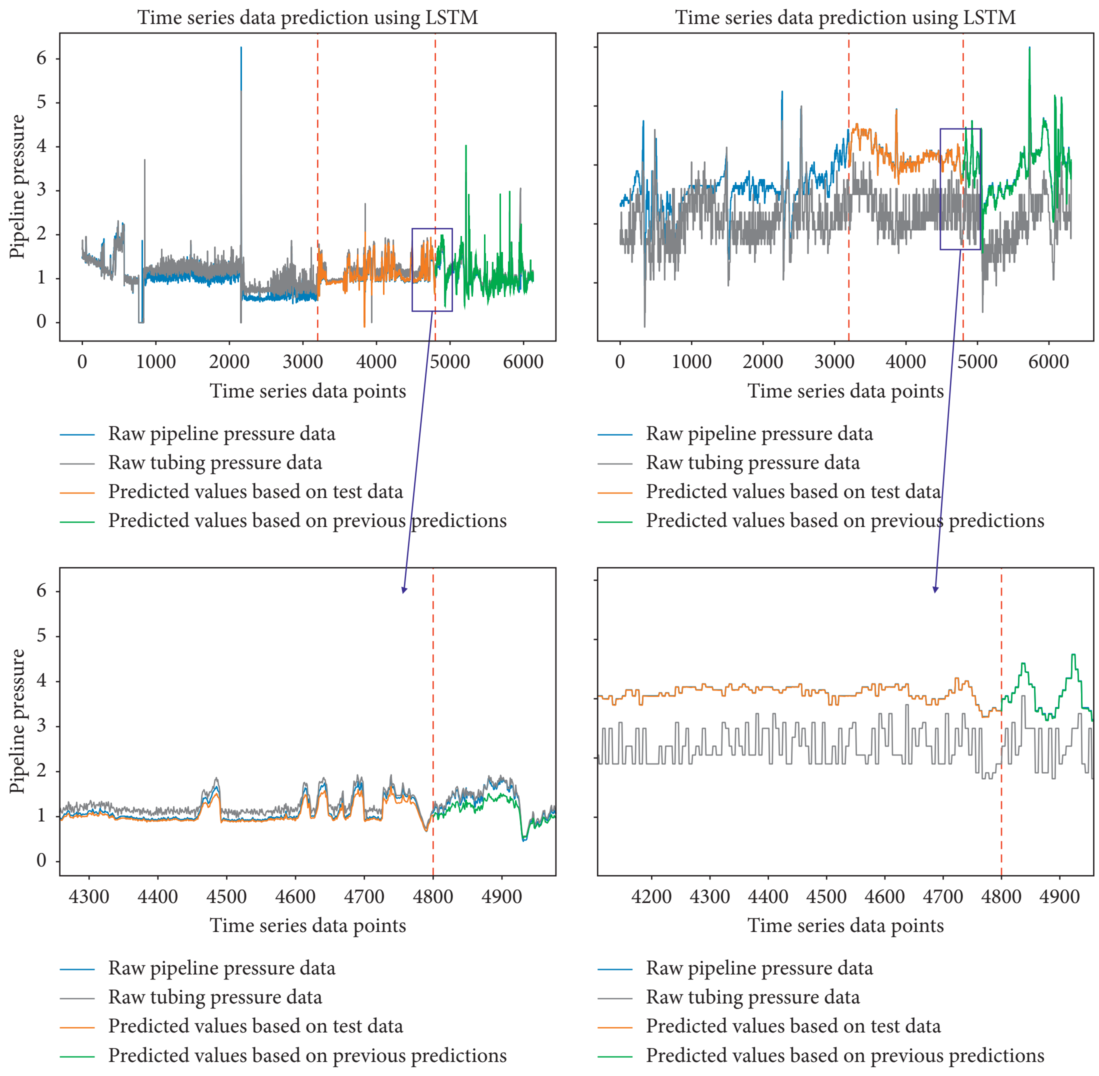

Figure 7: Predicted future trends of time-series data using LSTM.

training sequence length was set to 200, which means that the data of 200 time-steps before each sample point affects the value of the current point. The batch size was set to 200, and the number of training epochs was set to 100. Thus, approximately 32 groups of training data were randomly selected for each training session. In the validation process, the predictions from the second-dimensional test data were combined with the first-order difference data from the original first-dimensional data for future trend prediction.

5.3. Predictions and Evaluation of Accuracy. Two time-series datasets were used for model training and prediction to evaluate the accuracy of future trend predictions. The datasets are composed of oil-well tubing and pipeline pressure, using these two input variables and the first-order difference of the tubing pressure data to predict the future pipeline pressure trend. The predicted results are shown in Figure 7, wherein the blue and gray curves are the tubing and pipeline pressures. The orange curve is the predicted value

Table 3: Model prediction accuracy assessment.

Well name	Number of data points for prediction accuracy assessment	Mean absolute error	Average relative error (%)	Root mean square error
A1	(1312,3)	0.084	8.43	0.115
A2	(1312,3)	0.061	4.19	0.093
A3	(1312,3)	0.043	5.16	0.056
A4	(1312,3)	0.189	11.03	0.081

based on the test data, and the green curve is the predicted value based on the test data predictions joined with the original tubing pressure. As observed from the figure, LSTM can predict future trends. Moreover, the predicted value can be used to solve the problem of missing data owing to device failure or network interruptions.

A series of data from four additional wells were trained using our proposed model to evaluate the accuracy of predictions quantitatively. The mean absolute error (MAE), average relative error (ARE), and root-mean-square error (RMSE) were calculated to validate the model feasibility by comparing it with actual data. The results of the model accuracy assessment are presented in Table 2. We use the expected ARE to measure the prediction accuracy of the target actual data points in the experiment. As listed in Table 3, the ARE values range between 5% and 11%, and the RMSE values are generally sufficiently small. Hence, it can be observed that the prediction accuracy can meet practical requirements. It indicates that the proposed prediction method can achieve better prediction accuracy on different datasets.

6. Conclusions

In this study, we proposed the use of collaborative edge-cloud computing technologies in the IIoT system for efficient time-series data analysis, concerned different tasks, such as anomaly detection, time-series data future trend prediction, and missing data replacement. First, we adopted the use of machine learning algorithms—RRCF and isolation forest—to detect abnormal data point changes at the edge and prompt managers to pay attention to anomaly events. The random cut forest algorithm is more robust than the isolation forest algorithm. However, the latter has a higher sensitivity and more timely response time; thus, the combination of the two algorithms can deal more efficiently with anomaly situations. Second, the LSTM recurrent neural network algorithm was used in the cloud for future trend prediction and missing data replacement based on historical data. The differences between the predicted and the actual data values were compared using the RMSE to evaluate the accuracy of the predictions. The experimental results showed that the accuracy of predictions using the LSTM algorithm could fulfill practical applications.

There are still some limitations that need to be addressed and ongoing work for the RRCF algorithm and LSTM prediction model structure. At present, the simulations of the time complexity of the RRCF algorithm need to be improved. We will improve the simulation process of tree generation to adapt to deploy at the edge in future work. Besides, the stacked LSTM model for trend prediction in the cloud needs to be optimized and combined with a convolutional neural network to improve runtime efficiency and increase the prediction accuracy by 5%.

Future research can focus on the deployment of a lightweight database at the edge to expand local data storage and processing capabilities, increasing multi-source heterogeneous data access capabilities at the edge that can quickly parse data packets from the sensor layer to achieve remote management and predictive maintenance of devices, and on the improvement of the deep learning model and exploring integrated analysis and prediction using multiparametric data in the cloud to tap the value of time-series data.

Data Availability

The time-series data used to support the findings of this study have been deposited in the Gitee repository (https://gitee.com/mikefengshi/time-series-data-analysis/).

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The authors would like to thank Editage (http://www.editage.cn) for English language editing. This work was supported in part by the National Natural Science Foundation of China under Grant 61877041 and in part by Science and Technology Program of Chengdu under Grant 2020-GH02-00061-HZ.

References

[1] Y. Wu, Y. Liu, S. H. Ahmed, J. Peng, and A. A. Abd El-Latif, "Dominant data set selection algorithms for electricity consumption time-series data analysis based on affine transformation," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4347–4360, 2020.

[2] M. Heidari Kapourchali and B. Banerjee, "Unsupervised feature learning from time-series data using linear models," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3918–3926, 2018.

[3] J.-S. Fu, Y. Liu, H.-C. Chao, B. K. Bhargava, and Z.-J. Zhang, "Secure data storage and searching for industrial IoT by integrating fog computing and cloud computing," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4519–4528, 2018.

[4] A. Akbar, A. Khan, F. Carrez, and K. Moessner, “Predictive analytics for complex IoT data streams,” *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1571–1582, 2017.

[5] G. Premsankar, M. Di Francesco, and T. Taleb, “Edge computing for the Internet of Things: a case study,” *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1275–1284, 2018.

[6] J. Pan and J. McElhannon, “Future edge cloud and edge computing for Internet of Things applications,” *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 439–449, 2018.

[7] S. Ghosh, A. Mukherjee, S. K. Ghosh, and R. Buyya, “Mobi-IoST: mobility-aware cloud-fog-edge-IoT collaborative framework for time-critical applications,” *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 4, pp. 2271–2285, 2020.

[8] T. Wang, G. Zhang, A. Liu, M. Z. A. Bhuiyan, and Q. Jin, “A secure IoT service architecture with an efficient balance dynamics based on cloud and edge computing,” *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4831–4843, 2019.

[9] F. Hao, D.-S. Park, J. Kang, and G. Min, “2L.-M. C. ^{3}: 2L-MC 3: a two-layer multi-community-cloud/cloudlet social collaborative paradigm for mobile edge computing,” *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4764–4773, 2019.

[10] J. Tang, Z. Zhou, X. Xue, and G. Wang, “Using collaborative edge-cloud cache for search in Internet of Things,” *IEEE Internet of Things Journal*, vol. 7, no. 2, pp. 922–936, 2020.

[11] A. A. Cook, G. Misirli, and Z. Fan, “Anomaly detection for IoT time-series data: a survey,” *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6481–6494, 2020.

[12] M. Ceci, R. Corizzo, N. Japkowicz, P. Mignone, and G. Pio, “ECHAD: embedding-based change detection from multivariate time series in smart grids,” *IEEE Access*, vol. 8, pp. 156053–156066, 2020.

[13] Y. Liu, S. Garg, J. Nie et al., “Deep anomaly detection for time-series data in industrial IoT: a communication-efficient on-device federated learning approach,” *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6348–6358, 2021.

[14] M. Zhang, J. Guo, X. Li, and R. Jin, “Data-driven anomaly detection approach for time-series streaming data,” *Sensors*, vol. 20, no. 19, p. 5646, 2020.

[15] S. Dou, K. Yang, and H. V. Poor, “P. C. ^{2}A.: PC2A: predicting collective contextual anomalies via LSTM with deep generative model,” *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 9645–9655, 2019.

[16] E. Li, L. Zeng, Z. Zhou, and X. Chen, “Edge AI: on-demand accelerating deep neural network inference via edge computing,” *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 447–457, 2020.

[17] A. H. Sodhro, S. Pirbhulal, and V. H. C. de Albuquerque, “Artificial intelligence-driven mechanism for edge computing-based industrial applications,” *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 4235–4243, 2019.

[18] C. Savaglio and G. Fortino, “A simulation-driven methodology for IoT data mining based on edge computing,” *ACM Transactions on Internet Technology*, vol. 21, no. 2, pp. 1–22, 2021.

[19] Y. Ai, M. Peng, and K. Zhang, “Edge computing technologies for Internet of Things: a primer,” *Digital Communications and Networks*, vol. 4, no. 2, pp. 77–86, 2018.

[20] Y. Wang, Y. Shen, S. Mao, X. Chen, and H. Zou, “LASSO and LSTM integrated temporal model for short-term solar intensity forecasting,” *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2933–2944, 2019.

[21] Y.-F. Zhang, P. J. Thorburn, W. Xiang, and P. Fitch, “SSIM-A deep learning approach for recovering missing time series sensor data,” *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6618–6628, 2019.

[22] W. Yu, F. Liang, X. He et al., “A survey on the edge computing for the Internet of Things,” *IEEE Access*, vol. 6, pp. 6900–6919, 2018.

[23] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge computing: vision and challenges,” *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[24] C.-H. Chen, M.-Y. Lin, and C.-C. Liu, “Edge computing gateway of the industrial Internet of Things using multiple collaborative microcontrollers,” *IEEE Network*, vol. 32, no. 1, pp. 24–32, 2018.

[25] S. Guha, N. Mishra, G. Roy, and O. Schrijvers, “Robust random cut forest based anomaly detection on streams,” in *Proceedings of the The 33rd International Conference on Machine Learning*, pp. 2712–2721, ACM, New York, NY, USA, 19 June 2016.

[26] F. T. Liu, K. M. Ting, and Z. Zhou, “Isolation forest,” in *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, pp. 413–422, IEEE, NW Washington, DC. United States, 15 December 2008.