Hindawi
Wireless Communications and Mobile Computing
Volume 2020, Article ID 8897926, 10 pages
https://doi.org/10.1155/2020/8897926

Research Article

Anomaly Detection for Industrial Control System Based on Autoencoder Neural Network

Chao Wang,[1,2] Bailing Wang ,[1,2] Hongri Liu,[1,3] and Haikuo Qu[1,2]

[1]*School of Computer Science and Technology, Harbin Institute of Technology, Weihai, China 264209*
[2]*Research Institute of Cyberspace Security, Harbin Institute of Technology, Harbin, China 150001*
[3]*Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen, China 518000*

Correspondence should be addressed to Bailing Wang; wbl@hit.edu.cn

Received 23 April 2020; Revised 5 June 2020; Accepted 8 July 2020; Published 3 August 2020

Academic Editor: Fuhong Lin

As the Industrial Internet of Things (IIoT) develops rapidly, cloud computing and fog computing become effective measures to solve some problems, e.g., limited computing resources and increased network latency. The Industrial Control Systems (ICS) play a key factor within the development of IIoT, whose security affects the whole IIoT. ICS involves many aspects, like water supply systems and electric utilities, which are closely related to people's lives. ICS is connected to the Internet and exposed in the cyberspace instead of isolating with the outside recent years. The risk of being attacked increases as a result. In order to protect these assets, intrusion detection systems (IDS) have drawn much attention. As one kind of intrusion detection, anomaly detection provides the ability to detect unknown attacks compared with signature-based techniques, which are another kind of IDS. In this paper, an anomaly detection method with a composite autoencoder model learning the normal pattern is proposed. Unlike the common autoencoder neural network that predicts or reconstructs data separately, our model makes prediction and reconstruction on input data at the same time, which overcomes the shortcoming of using each one alone. With the error obtained by the model, a change ratio is put forward to locate the most suspicious devices that may be under attack. In the last part, we verify the performance of our method by conducting experiments on the SWaT dataset. The results show that the proposed method exhibits improved performance with 88.5% recall and 87.0% F1-score.

1. Introduction

In the context of Industry 4.0, the Industrial Internet of Things (IIoT) has attracted high attention in the academia and industry. Within IIoT, more and more devices have been joined together and produce massive industrial data every day, which requires powerful computing resources. Benefited from cloud computing, enterprises can move the computing tasks into the cloud instead of their own physical machines in order to mitigate the pressures. Also, cloud computing can be integrated into the mobile computing environment, which is called mobile cloud computing.

However, as the development of IIoT, the amount of data collected from sensors or other devices grows exponentially. When these data are transmitted to the cloud, network latency and bandwidth become a bottleneck [1]. To overcome these problems, edge computing (e.g., cloud-aware fog computing mechanism [2, 3]) is one of the most promising solutions. In real applications, industrial enterprises could make some necessary computing tasks close to the machine in the industrial control system (ICS). Only some important computing tasks or results are delivered and stored in the cloud center [4]. Workloads of the cloud center could be reduced sharply in this way. There are some researches that focus on problems of edge computing. Since data are sent to remote machines, the security of data transmission is another problem; thus, the authors of [5] proposed a new secure communication scheme to solve it.

In fact, ICS is an important part of the industrial edge, and its safety issues are becoming critical to the IIoT's development. Attacks on ICS have increased over the last two decades, and the most famous one is Stuxnet. Damages to

ICS can cause serious consequences; therefore, the study of methods to protect these systems is very important.

Intrusion detection system (IDS) is one means to provide protection. There are two ways to classify IDS according to their techniques used, namely, signature-based IDS and anomaly-based IDS [6]. When using signature-based techniques, the attacks are detected by comparing the characteristics of known attacks with new events like traffic and commands. An anomaly-based IDS constructs a template of normal behavior and detects attacks by calculating the deviations of observed behavior with the template. Since anomaly detection need only normal work conditions to learn the normal profile, it can detect unknown attacks. We focus on anomaly-based techniques in this paper.

There are two main issues concerned with anomaly detection of ICS. Firstly, it is a challenging task to model the complex system. Since there are many devices within the system, it is hard to learn the associations between them. How to locate abnormal devices is another challenge. Methods finding the devices that work abnormally could help workers to check the system in time correctly, which could reduce losses caused by anomalies.

Many anomaly detection techniques have been specifically developed for ICS. A discrete multi-input and multioutput (MIMO) system model [7] is used to represent the control and process behavior. But the model is not applicable to the nonlinear model. Using the phenomenon that traffic between devices is periodic; Goldenberg and Wool [8] modeled the Modbus/TCP traffic by deterministic finite automaton (DFA). However, their model is suitable for single-period traffic patterns only. Deep learning has demonstrated promising results to learn the complicated relations of variables. There are some works using deep learning methods to do anomaly detection [9, 10]. In order to achieve the anomaly detection in ICS, we use a composite autoencoder model similar to [11] to learn the work pattern of ICS and with our contributions as follows.

(i) We propose a composite autoencoder model to learn the work pattern of ICS by predicting and reconstructing the input data. Anomaly is detected using the error obtained by the model

(ii) Using the error distribution, we can locate the variables that behave abnormally. We define a change ratio to seek which devices are suffering attack

(iii) We conduct experiments on the SWaT dataset, which is collected from a real ICS. Experiment results show that our method outperforms the other three methods with 88.5% recall and 87.0% F1-score

The remainder of this paper is organized as follows. Some related work about anomaly detection is summarized in Section 2. Section 3 introduces the dataset used in this paper briefly. The problem of detecting anomaly dealt with time series data is analyzed theoretically in Section 4. Section 5 describes our proposed method in detail. Section 6 conducts the experiments as well as performance analysis. Lastly, Section 7 presents our conclusions and future work.

2. Related Work

The security issues of ICS have been extensively studied. In [12], the authors presented some threats and secure methods for these infrastructures. In [13, 14], the authors surveyed some researches on ICS and also presented some challenges that need to be addressed. Two strategies were described in [15] for securing SCADA networks in general. Besides, many researchers have put forward new techniques based on anomaly detection.

Anomaly detection method discovery attacks by estimating its differences with the normal profile. The normal profile can be constructed using many categories of data sources. The work in [8, 16–18] used network traffic within ICS as a data source to model the normal communication. The Hidden Markov model (HMM) [16] was used to model packets delivered between devices for intrusion detection. In addition, [17] proposed a method that learns the Modbus/TCP traffic transactions using the request message only. The authors of [18] employed a dynamic Bayesian network structure to characterize normal command and data sequences at a network level and achieved a low false positive rate.

Every device in ICS are assumed to have their behavior pattern, the authors of [19] used features like the data response time and the physical operation time to create a physical fingerprint for the devices within ICS environment. To find malicious codes, a deep learning method was utilized to model the normal behavior of the power-grid controller [20].

In this research, we aim to use the time series data of devices within ICS to model the normal working condition. When it comes to anomaly detection of multivariable time series, some deep learning methods have been put forward. Malhotra et al. [9] used the stacked LSTM neural network. The network is trained on normal data only, and the prediction errors are used to determine whether the observation is normal or anomalous. And the mechanism in [10] reconstructed the normal time series data with the LSTM encoder-decoder. In [21], a deep convolutional neural network (CNN) is used to predict the time series.

Our proposed method experiments on the SWaT dataset would be described in Section 3. There are some works that have been done on this dataset. Goh et al. [22] used the LSTM neural network to predict the time series of ICS and used cumulative sum (CUSUM) for anomaly detection, but only the P1 stage in the dataset had been checked. The timed automata learning is combined with the Bayesian network for anomaly detection [23]. Inoue et al. [24] proposed two methods to detect the anomalies, namely, DNN and one-class SVM. In this research, the work in [23, 24] is used to compare with our proposed method.

3. Dataset Description

In this paper, we use the Secure Water Treatment (SWaT) dataset to test and verify our method. The SWaT dataset is provided by the Singapore University of Technology and Design [25]. It is collected from a water treatment plant testbed that produces purified water. Within the testbed,

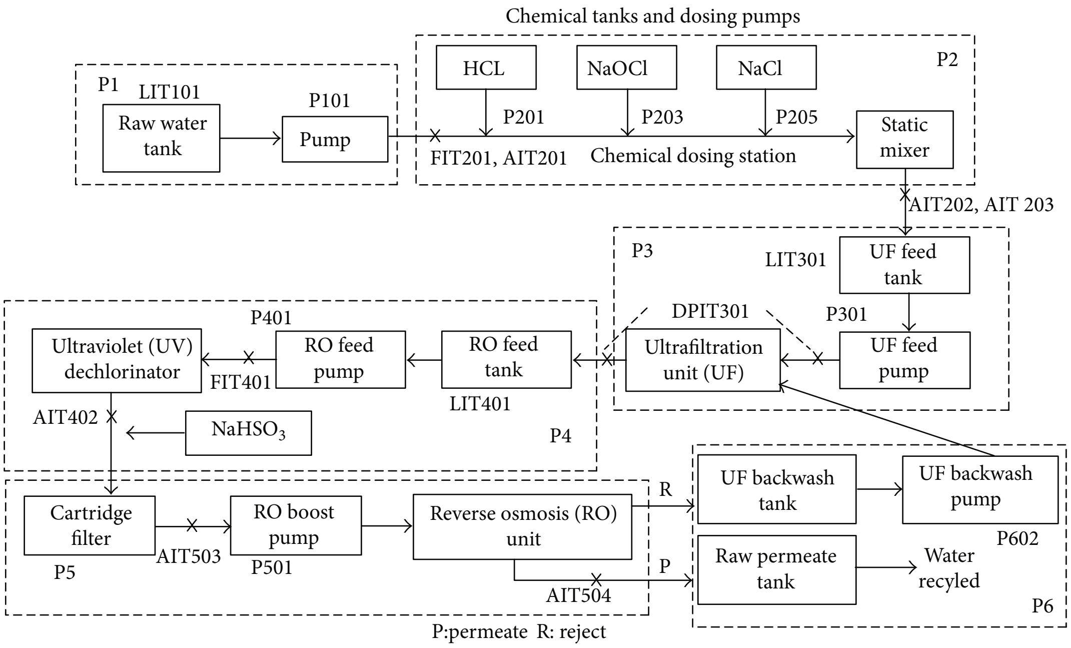

Figure 1: The SWaT testbed process overview from [25].

Table 1: Description of attack types.

Attack type	Meaning	Example
Single Stage Single-Point (SSSP)	Attack to one device in any single stage	The 1st attack with target MV-101
Single Stage Multi Point (SSMP)	Attack to multiple devices in any single stage	The 16th attack with targets MV-101 and LIT-101
Multi Stage Single Point (MSSP)	Attack to one devices in many stages	The 33rd attack with targets AIT-402 and AIT-502
Multi Stage Multi Point (MSMP)	Attack to multiple devices in many stages	The 18th attack with targets P-602, DPIT-301, and MV-302

there are a numbers of devices, which are categorized into a sensor and actuator. The devices are distributed in six stages as demonstrated in Figure 1.

All the data of sensors and actuators are logged every second during the total 11 days of running, which means it is a multivariable time series data. In the whole running, the first seven days were under normal condition. And there are 36 attacks in the remaining four days. In the attack data, there are four types of attacks, namely, SSSP, SSMP, MSSP, and MSMP, which are described detailed in Table 1. For full explanations of the dataset, please refer to [25].

The SWaT dataset contains traffic data also, which had been parsed already. But in this paper, the time series data are our focus of work.

4. Problem Statement

As described in Section 3, the data we deal with are multiple variable time series. Consider a time series $X = \{x_1, x_2, \cdots, x_N\}$ of length N, where one point is a m-dimensional vector $\{x_t^1, x_t^2, \cdots, x_t^m\}$ at time $t (t \leq N)$. We use a window of length T sliding over the time series to obtain multiple time series. Also, a sequence of continuous observations $\{x_t, x_{t+1}, \cdots, x_{t+T}\}$ from time t to $t + T$ is denoted as $x_{t:t+T}$.

The objective of anomaly detection for multivariable time series is to find anomalous part exploiting the regular pattern appeared in the history data.

In this research, we use the observations under normal working condition to train the model for learning the system work pattern by predicting future time series and reconstructing the origin input. After model learning the working pattern, it is used to detect the test dataset which includes attacks. The anomaly results are obtained by comparing the error of attack data against a threshold. A higher error indicates the point is anomalous with a higher likelihood.

5. Proposed Method

The whole process of our proposed method consists of two parts, training phase and testing phase. The training part learns the working pattern of ICS using only normal data. When the model is trained, the error obtained by training on normal data is used to select ananomaly threshold. The testing part checks the model's performance using the data that includes attacks. The trained model is used to reconstruct and predict the testing data that includes attacks. The observations whose error is higher than the threshold are denoted as anomalous. After that, the abnormal part which may be suffering an attack is located. The whole structure of our method is demonstrated in Figure 2. The detail of our model is introduced in this section. In the process of learning the pattern of ICS, we employed a composite autoencoder model, which has the power to learn nonlinear relationships. Also, a change ratio is proposed to locate the anomaly devices.

5.1. Autoencoder Model. Researchers from different scientific fields have applied deep learning in their respective area of research, since deep learning has exhibited a powerful ability to solve complex problems in fields like computer vision and natural language processing in recent years.

An autoencoder neural network is an unsupervised learning framework. Generally speaking, an autoencoder has an input layer, an output layer, and one or more hidden layers. Unlike common deep neural networks, an autoencoder has an architecture where hidden layers are smaller than input layers. Benefited from this, it could learn a compressed representation.

In this research, we use the architecture of the autoencoder as shown in Figure 3, which is divided into two parts, the encoder and the decoder. The compressed representation of input data is learned by the encoder part, from which the decoder reconstructs the input. In some researches, after training the autoencoder, the decoder is removed and the remaining part is used for classification. However, we use the whole autoencoder to reconstruct the origin input.

We choose LSTM as the building blocks for the autoencoder model to deal with the time series. LSTM is a special RNN (recurrent neural network); it is designed to solve problems suffered by common RNN. Due to space limitations, we do not expand a detailed description.

5.2. Composite Model. The LSTM autoencoder can reconstruct the input (see in Section 5.1). It can be used as a future predictor also. In this paper, we use a composite autoencoder to achieve the goals of the learning the pattern of ICS normal working. The designed model can do the input reconstruction and future prediction work both, which can learn better data representations using them both in the same time [11]. Its architecture is shown in Figure 4.

Imagine the input of a composite model is $x_{t:t+T}$, whose length is T. The model could output two part data, namely, the reconstructed time series and the predicted time series. We use y_t and z_t to denote the output of the model. Considering a time series $x_{t:t+T}$ as the input of the composite model, $y_{t:t+T}$ is the reconstructed series and $z_{t+T:t+2T}$ is the predicted series. For example, an input time series of length 5, $\{x_0, x_1, x_2, x_3, x_4\}$. The output of the model is in two parts, the reconstruction part $\{y_0, y_1, y_2, y_3, y_4\}$ and prediction part $\{z_5, z_6, z_7, z_8, z_9\}$, separately.

The MSE (mean square error) is used to calculate the difference between the actual input and output. Considering an input series x_t, its mth dimension value is x_t^m, and the prediction or reconstruction output is $\widehat{x}_t^m$. The calculation of MSE is given in (1).

$$\mathrm{MSE} = |x_t^m - x\wedge_t^m|^2 \tag{1}$$

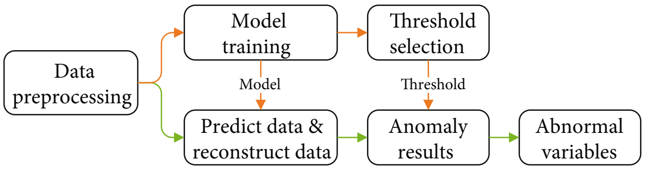

FIGURE 2: The whole structure of the proposed anomaly detection method.

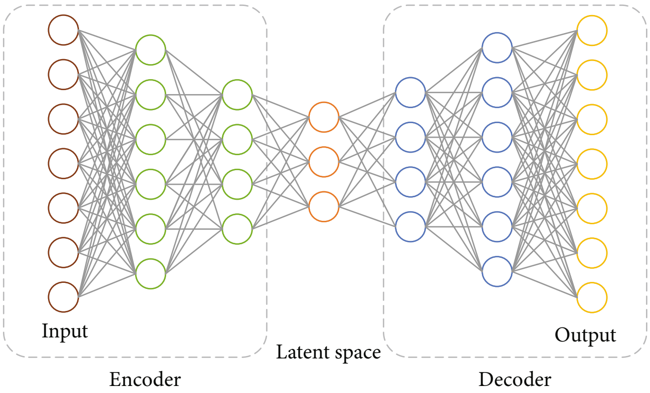

FIGURE 3: An example of autoencoder neural network.

And the loss function L is used to calculate MSE of the whole time length. The reconstruction part loss is L_c and the prediction part loss is L_p. The model will try to minimize the sum of both parts, $L_s = L_c + L_p$.

5.3. Anomaly Detection. In this section, we introduce the process of anomaly detection. As we mentioned in the previous section, the two-part output of composite autoencoder model aren't matched in the time dimension. The $y_{t:t+T}$ is reconstructed from input $x_{t:t+T}$, and the $z_{t:t+T}$ is predicted from input $x_{t-T:t}$. It is significant to adjust the output values and make them match in time dimension before it passes to anomaly detection.

The error obtained by two parts is calculated separately and added together to obtain the overall error value E_t. The steps of calculating are as follows.

$$e_t^{\text{cons}} = |x_t - y_t|, \tag{2}$$

$$e_t^{\text{pred}} = |x_t - z_t|, \tag{3}$$

$$E_t = e_t^{\text{cons}} + e_t^{\text{pred}}. \tag{4}$$

In order to eliminate the error that happened due to abruptly changing of the input data, we use an exponentially weighted moving average method (EWMA) like paper [26] to obtain a smoothed error SE_t.

$$\mathrm{SE}_t = \alpha E_t + (1-\alpha)\mathrm{SE}_{t-1}, \tag{5}$$

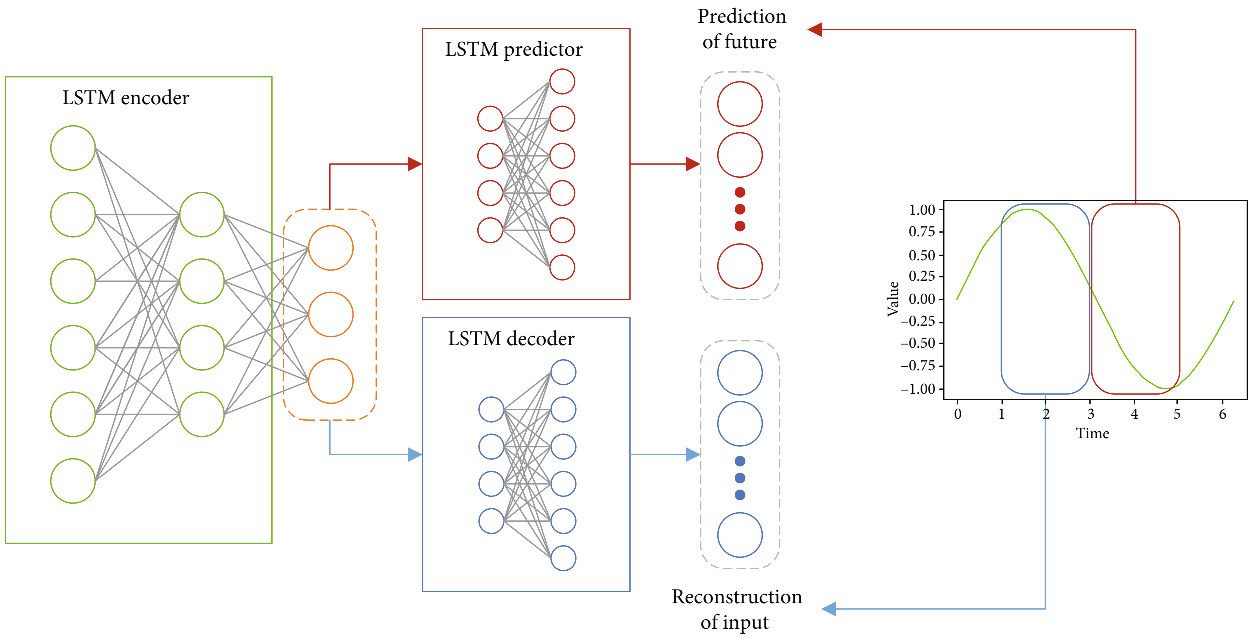

Figure 4: A composite autoencoder model.

$$\mathrm{SE}_0 = 0, \tag{6}$$

$$\alpha = 1 - \exp^{\frac{\ln(0.5)}{H}}. \tag{7}$$

Meanwhile, we use a power technique to decrease the false negative rate.

$$\mathrm{PE}_t = \frac{1}{m}\sum_{i=1}^{m}\left|\mathrm{SE}_t^i\right|^p. \tag{8}$$

When the error PE_t is higher than threshold V_{thre}, the observation at time t is classified as anomaly. The threshold V_{thre} is selected from the error PE_t of training part. Steps used to detect anomaly are listed in Algorithm 1.

$$\mathrm{A}_t = \begin{cases} 1, & \mathrm{PE}_t \geq V_{\text{thre}}, \\ 0, & \text{Else}. \end{cases} \tag{9}$$

5.4. Locate Attacked Variables. Generally speaking, with the purpose of damaging the ICS or affecting its normal work, some attackers choose to destroy the devices like sensors or actuators. After the method mentioned in the last section detecting anomaly successfully when attacks happen, the next step is to locate variables (devices in ICS) that are suffering an attack. It helps the workers find anomaly and take measures to bring the system back to normal.

Unlike research [26] using the greatest error of variables, we use a change ratio to denote whether one part is anomalous. We believe if one device is attacked by attackers, the SE_t obtained within the anomaly detection phase presents difference between the normal and attacked conditions. Specifically, our model should output lower SE_t when ICS works normally than the time when the attacks happen. Under this assumption, the change ratio is proposed to measure the difference. The change ratio is given in (10).

```
Input:
    Input time series $x_t$; reconstruct time series $y_t$;
    Predict time series $z_t$; error check threshold $V_{\text{thre}}$;
Output:
    Anomaly result $A_t$;
1: Calculate sum of error $E_t$ by (2), (3), (4)
2: Calculate smoothed error $\mathrm{SE}_t$ by EWMA methods (5)
3: Calculate $p$-power error $\mathrm{PE}_t$ by (8)
4: Mark the anomaly $A_t$ by (9)
5: Return the anomaly observations $A_t$
```

Algorithm 1: Anomaly detection.

$$c_t^i = \frac{1}{T_c}\left|\mathrm{SE}_{t:t+T_c}^i - \mathrm{SE}_{t:t-T_c}^i\right|. \tag{10}$$

When one observation x_t at time t is classified as anomaly, every dimension of its SE_t is checked by calculating the difference between average of SE_t^i on time window before the attack and after it. The length of the time window is denoted as T_c. In this research, some attacks among test dataset have multiple targets. Therefore, we use the top-k to denote k devices that are under attack.

6. Experiments and Result Analysis

Based on the method mentioned in above part, the process of experiments on the SWaT dataset and results are listed in this section. First, we demonstrate the preprocessing steps for the input data. After that, the machine environment where our

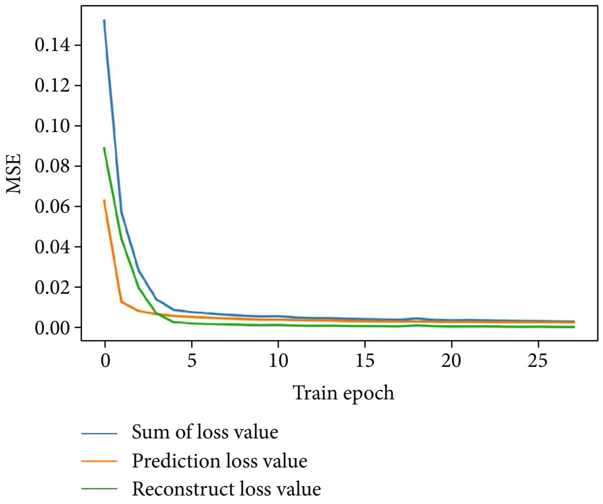

Figure 5: Training loss.

model is trained and the evaluation metrics we used to access our method are introduced. In the last part, we analyze the final results in detail, including attack detection results, the comparisons with other methods, etc.

6.1. Data Preprocessing. There are 51 variables (devices) in the SWaT dataset. The training data has 496,800 records which are collected when the testbed works under a normal condition. Also, testing data has 449,919 records and there are 36 attacks among them. Because the system has to be stabilized, so the first part of normal data has been trimmed.

In the experiment, we find that some variables are unstable, e.g., AIT-201. Its distribution between train dataset and test dataset is different extremely. Since we use normal data to train the model only, so these unstable and unrepresentative variables are removed, including AIT-201, P201, and all the variables in P6. Especially, variables in the P6 stage were not completely used during data collection as said in [23]. After removing these variables, 45 variables are remaining.

In this research, we use a sliding window approach to divide origin time series, and the length of window is l. In order to learn the whole pattern in the training phase, an overlap length l_{overlap} is used. Overlap means two continuous time series have the same part, the beginning l_{overlap} length of the second part is the same as the end l_{overlap} of the first part. But when calculating the training error to select the threshold or testing the model, the data are divided without overlap.

To accelerate the training speed and increase detect accuracy, all the training data are scaled to (0, 1). It would be wrong if we scale all the data including test data, since information would be leaked in this way. The test set that includes the attack is scaled using the minimal and maximal of every variables from the training set. Because our method uses a composite model, the output constructed part and predicted part are not in same time window, which result in the first sequence having only the reconstruction part and the last sequence do not have a prediction part. For these circumstances, the error only count one part.

Table 2: Confusion matrix.

True class	Detection result Attack	 Normal
Attack	True positive (TP)	False negative (FN)
Normal	False positive (FP)	True negative (TN)

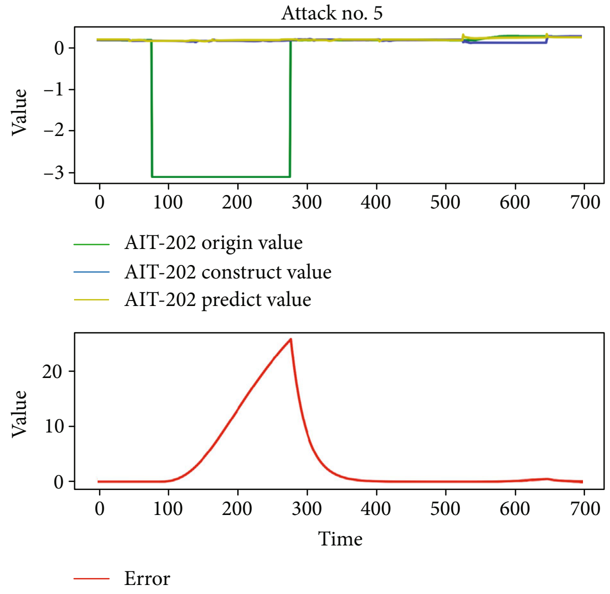

Figure 6: Detection result of attack no. 5.

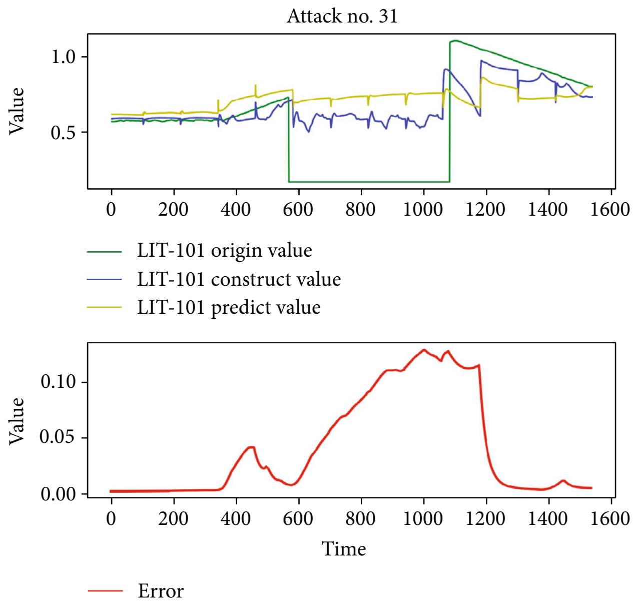

Figure 7: Detection result of attack no.31.

In the experiment, we set l as 120 seconds and overlap length l_{overlap} as 115 seconds. Although the overlap part of every time series is large, we believe the model learns better

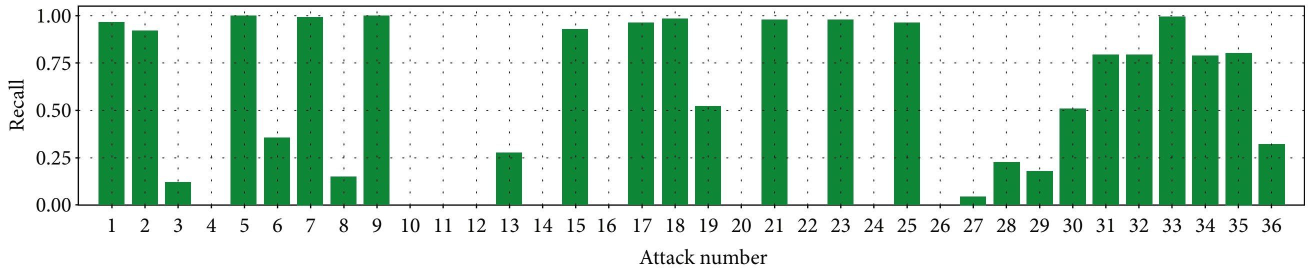

Figure 8: Recall of all attacks.

Table 3: Comparison of the performance with other methods.

Method	Precision	Recall	F1 score
DNN	0.98295	0.67847	0.80281
SVM	0.92500	0.69901	0.79628
TABOR	0.86171	0.78803	0.82322
Com-AE	0.85596	0.88525	0.86976

representation using more time series by such allocation. Some arguments used in the method are listed here. The H used in (6) is 120 seconds, which has the same length with l. The p is 4 and the T_c for locating suspicious variables is 120 seconds also. The maximum of error obtained by training on the nonoverlap time series is selected as threshold.

6.2. Model Training. The model is trained and tested on our machine which consists of Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz, 3 NVIDIA Tesla P100 PCIe 12GB, and 128 GB RAM.

We implement our model based on Keras library (version 2.2.4) whose programming of neural networks is more convenient than Tensorflow. In order to elevate the training and testing time, we use the CuDNNLSTM as the building block, which has the same effects with LSTM.

In the experiment, the neuron number of two layers within encoder part is 64 and 32, respectively. The first layer is larger than the input dimension, which does not strictly obey the rule of hidden layers is smaller than the input layer. But it gets better performance in the experiments still. The neuron number of reconstruction part and prediction part is 32 and 64, which are symmetry with the encoder part.

We choose the Adam optimizer [27] and the training epochs is 100. Earlystop is used to cut down the training time. To overcome the problem of local minimum when training neural network, we train the composite autoencoder multiple times and choose the best one to present. The training losses of both parts are shown in Figure 5.

6.3. Evaluation Metrics. In the dataset, the records are labeled as "attack" or "normal" for every second. We use this label directly to evaluate our detect results. All possible results are listed in Table 2. When one record is classified as attack if it is attack indeed, this is a true positive (TP). Otherwise, a false negative (FN) is the situation when an attack record is marked as normal. Also, a true negative (TN) is that normal records been classified correctly. For the last one, a false positive (FP) is a prediction that the normal record is misclassified as an attack. We use $\text{Precision} = \text{TP}/\text{TP} + \text{FP}$, $\text{Recall} = \text{TP}/\text{TP} + \text{FN}$, and $\text{F}_1 = (2 * \text{Precision} * \text{Recall})/(\text{Precision} + \text{Recall})$ to denote our performance.

6.4. Result Analysis. Before diving into the overall performance, we first analyze the detected results of some attacks. We demonstrate our method using two examples, attack no. 5 and attack no. 31.

In Figure 6, the original value is presented with the construction part and prediction part. Its value was decreased to a lower value during the attack. Since we use the EWMA method to smooth the error, the shape of the error is not a rectangle. We detect this attack with 100% recall. With decreasing the sensor's value to extremely lower value, the error obtained after p-power processing is bigger. As a result, this attack is easily detected relatively.

Another attack is no. 31, which impacts device LIT-101, as shown in Figure 7. Compared with attack no. 5, it causes lower value change. The value of error is small also, which affects the detect performance. The recall for this attack is 79.6%.

From Figures 6 and 7, we can see that the error does not shrink quickly after the attack is over. This is because the system needs time to stabilize. The recall of all attacks is shown in Figure 8.

We compare the results of our method (Com-AE) with the other three methods, which are DNN and SVM [24], as well as TABOR [23]. The performance obtained is shown in Table 3. Our method achieves higher recall and F1-score when compared with the other three methods. But the precision is lower. Through analysis, we find that using a higher threshold will obtain a higher precision. The number of detected attacks is reduced in this way, however. With consideration of detecting more attacks, we use the maximal error obtained during the training phase as the threshold as described before.

A detailed comparison of recall for every attack is listed in Table 4. There are 36 attacks total within testing data which were the records in the remaining 4 days. DNN and SVM fail to check some attacks. TABOR method makes 24 attacks detected. Our method Com-AE detects 26 attacks successfully. It seems some attacks may have little impact on the system since all the methods fail to detect them.

Table 4: Comparisons with other models. The results of TABOR are taken from paper [23]. And the results of SVM and DNN are taken from [24].

No.	Target	Detection	Recall			
			DNN	SVM	TABOR	Com-AE
1	MV-101	FIT-101, **MV-101**, PIT-502	0	0	0.049	0.966
2	P-102	**P-102**, P-302, FIT301	0	0	0.930	0.919
3	LIT-101	P-302, FIT-301, DPIT-301	0	0	0	0.120
4	MV-504	—	0	0.035	0.328	0
5	AIT-202	**AIT-202**, P-203, P-302	0.717	0.720	0.995	1.0
6	LIT-301	P-205, P-203, P101	0	0.888	0	0.354
7	DPIT-301	P-302, **DPIT-301**, FIT-301	0.927	0.919	0.992	0.992
8	FIT-401	**FIT-401**, UV-401, P-501	1	0.433	0.994	0.150
9	FIT-401	UV-401, **FIT-401**, P-501	0.978	1	0.998	1.0
10	MV-304	—	0	0	0	0
11	MV-303	—	0	0	0	0
12	LIT-301	—	0	0	0	0
13	MV-303	P-101, **MV301**, P-302	0	0	0.597	0.275
14	AIT-504	—	0.123	0.13	0.004	0
15	AIT-504	**AIT-504**, AIT-503, AIT-501	0.845	0.848	0.997	0.929
16	MV-101,LIT-101	—	0	0.0167	0.083	0
17	UV-401,AIT-502,P-501	**UV-401**, **P-501**, FIT-504	0.998	1	0.998	0.961
18	P-602,DPIT-301,MV-302	**DPIT-301**, P-302, FIT-301	0.867	0.875	0	0.984
19	P-203,P-205	**P-203**, **P-205**, P-204	0	0	0	0.523
20	LIT-401,P402	—	0	0.009	0	0
21	P-101,LIT-301	P-102, AIT-503, FIT-201	0	0	0.999	0.976
22	P-302,LIT-401	—	0	0	0.196	0
23	P-302	**P-302**, MV-304, FIT-301	0.936	0.936	1.000	0.977
24	P-201,P-203,P-205	—	0	0	0	0
25	P-101,MV-101,LIT-101	FIT-201, P-102, AIT-503	0	0.003	0.999	0.964
26	LIT-401	—	0	0	0	0
27	LIT-301	P-205, P-203, P-101	0	0.905	0	0.045
28	LIT-101	P-302, FIT-301, DPIT-301	0	0	0.890	0.225
29	P-101	P-102, **P-101**, AIT-503	0	0	0.990	0.178
30	P-101,P-102	AIT-202, **P-101**, FIT-201	0	0	0.258	0.507
31	LIT-101	AIT503, MV304, **LIT-101**	0	0.119	0.889	0.796
32	P-501,FIT-502	FIT-504, FIT-503, PIT-502	1	1	0.998	0.795
33	AIT402,AIT502	FIT-101, **AIT-502**, **AIT-402**	0.923	0.927	0.996	1.0
34	FIT-401,AIT-502	AIT-503, **FIT-401**, **AIT-502**	0.940	0	0.369	0.788
35	FIT-401	**FIT-401**, P-501, UV-401	0.933	0.927	0.997	0.805
36	LIT-301	**LIT-301**, AIT-402, AIT-502	0	0.357	0	0.321

6.5. Detected Target. Because of space limitation, we just list attacks associated with devices within the P1 stage in Table 5. As Table 5 shows, our method can locate 5 attacks within 10 attacks rightly. Even if there are 5 attacks located, they are not recognized as the most suspicious anomaly part.

There are some reasons behind these. The variables are associated with each other closely since ICS is a complex system. And this system includes two types of devices, namely, the sensor and actuator. When one variable is suffering an attack, the other one would change also. The degree may be higher than the original variables. How to detect the most suspicious part considering these factors will be our future work directions.

For the detailed detection result, please check Table 4, where all the total 36 attack detection result are listed.

7. Conclusions and Future Work

A composite autoencoder neural network model is proposed in this paper. The model is aimed at learning the pattern of

Table 5: Detection results in P1.

No	Target	Detection
1	MV-101	FIT-101, **MV-101**, P-502
2	P-102	**P-102**, P-302, FIT301
3	LIT-101	P-302, FIT-301, DPIT-301
16	MV-101,LIT-101	—
21	P-101,LIT-301	P-102, AIT-503, FIT-201
25	P-101,MV-101,LIT-101	FIT-201, P-102, AIT-503
28	LIT-101	P-302, FIT-301, DPIT-301
29	P-101	P-102, **P-101**, AIT-503
30	P-101,P-102	AIT-202, **P-101**, FIT-201
31	LIT-101	AIT503, MV304, **LIT-101**

ICS working conditions using data under normal conditions only. After predicting and reconstructing the origin input time series at the same time, the error obtained from both parts is used to determine whether an observation of multivariable time series is anomalous or normal. With anomaly results, the change ratio comes to locate the variables which are suffering an attack.

We demonstrate the effectiveness of our methods by experimenting on the SWaT dataset which is collected from a real industrial control system. The F1 score is 87.0% and recall is 88.5%, which is higher than other current researches.

In the future, there are some directions to enhance the performance of our method.

(i) When handling the error obtained by the neural network model, a static value is selected as a threshold to determine whether an anomaly happens. It will be more accurate if we use a dynamic threshold

(ii) Although the algorithm used for locating attacked variables can find target most likely in some attack scenes. The accuracy needs some improvements

Also, we have only experimented with our method on the SWaT dataset. It is more significant to test its performance on more datasets.

Data Availability

Readers who want to reproduce our result or test their own methods can access the dataset used in this research from the website: https://itrust.sutd.edu.sg/itrust-labs_datasets. Please follow the instructions on the website to obtain this dataset.

Conflicts of Interest

All the authors hereby declare no conflicts of interest.

Acknowledgments

We thank the iTrust, Centre for Research in Cyber Security, Singapore University of Technology and Design for designing and sharing the SWaT dataset. This research is funded by the National Key Research and Development Program of China (No. 2018YFB2004200).

References

[1] F. Song, Z. Ai, Y. Zhou, I. You, K. R. Choo, and H. Zhang, “Smart collaborative automation for receive buffer control in multipath industrial networks,” *IEEE Transactions on Industrial Informatics*, vol. 16, no. 2, pp. 1385–1394, 2020.

[2] R. Menchaca-Mendez, B. Luna-Nuez, R. MenchacaMendez, A. Yee-Rendon, R. Quintero, and J. Favela, “Opportunistic mobile sensing in the fog,” *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 2796282, 18 pages, 2018.

[3] C. Gong, F. Lin, X. Gong, and Y. Lu, “Intelligent cooperative edge computing in the internet of things,” *IEEE Internet of Things Journal*, p. 1, 2020.

[4] F. Song, M. Zhu, Y. Zhou, I. You, and H. Zhang, “Smart collaborative tracking for ubiquitous power iot in edge-cloud interplay domain,” *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6046–6055, 2020.

[5] H. Hui, C. Zhou, S. Xu, and F. Lin, “A novel secure data transmission scheme in industrial internet of things,” *China Communications*, vol. 17, no. 1, pp. 73–88, 2020.

[6] I. N. Fovino, “SCADA system cyber security,” in *Secure Smart Embedded Devices, Platforms and Applications*, pp. 451–471, Springer, New York, NY, USA, 2014.

[7] S. Zhanwei and L. Zenghui, “Abnormal detection method of industrial control system based on behavior model,” *Computers & Security*, vol. 84, pp. 166–178, 2019.

[8] N. Goldenberg and A. Wool, “Accurate modeling of modbus/tcp for intrusion detection in SCADA systems,” *International Journal of Critical Infrastructure Protection*, vol. 6, no. 2, pp. 63–75, 2013.

[9] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, “Long short term memory networks for anomaly detection in time series,” *European Symposium on Artificial Neural Networks*, i6doc, 2015.

[10] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, “LSTM-based encoder-decoder for multi-sensor anomaly detection,” in *ICML 2016 Anomaly Detection Workshop*, New York, NY, USA, July 2016.

[11] N. Srivastava, E. Mansimov, and R. Salakhutdinov, “Unsupervised learning of video representations using LSTMs,” in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37, ICML15*, pp. 843–852, Lille, France, 2015.

[12] S. Plaga, N. Wiedermann, S. D. Anton, S. Tatschner, H. Schotten, and T. Newe, “Securing future decentralised industrial IoT infrastructures: challenges and free open source solutions,” *Future Generation Computer Systems*, vol. 93, pp. 596–608, 2019.

[13] M. Krotofil and D. Gollmann, “Industrial control systems security: what is happening?,” in *2013 11th IEEE International Conference on Industrial Informatics (INDIN)*, pp. 670–675, Bochum, Germany, July 2013.

[14] M. Kaouk, J. Flaus, M. Potet, and R. Groz, “A review of intrusion detection systems for industrial control systems,” in *2019 6th International Conference on Control, Decision and Information Technologies (CoDIT)*, pp. 1699–1704, Paris, France, April 2019.

[15] R. Chandia, J. Gonzalez, T. Kilpatrick, M. Papa, and S. Shenoi, "Security strategies for SCADA networks," in *Critical Infrastructure Protection. ICCIP 2007. IFIP International Federation for Information Processing, vol 253*, E. Goetz and S. Shenoi, Eds., pp. 117–131, Springer, Boston, MA, USA, 2008.

[16] K. Stefanidis and A. G. Voyiatzis, "An HMM-based anomaly detection approach for SCADA systems," in *Information Security Theory and Practice. WISTP 2016. Lecture Notes in Computer Science, vol 9895*, S. Foresti and J. Lopez, Eds., pp. 85–99, Springer, Cham, Heraklion, Greece, 2016.

[17] B.-K. Kim, D.-H. Kang, J.-C. Na, and T.-M. Chung, "Detecting abnormal behavior in scada networks using normal traffic pattern learning," in *Computer Science and its Applications. Lecture Notes in Electrical Engineering, vol 330*Springer, Berlin, Heidelberg.

[18] M. K. Yoon and G. Ciocarlie, "Communication pattern monitoring: improving the utility of anomaly detection for industrial control systems," in *Proceedings 2014 Workshop on Security of Emerging Networking Technologies*, San Diego, CA, USA, 2014.

[19] D. Formby, P. Srinivasan, A. Leonard, J. Rogers, and R. Beyah, "Who's in control of your control system? Device fingerprinting for cyber-physical systems," in *Proceedings 2016 Network and Distributed System Security Symposium*, San Diego, CA, USA, Febuary 2016.

[20] Z. He, A. Raghavan, G. Hu, S. Chai, and R. Lee, "Power-grid controller anomaly detection with enhanced temporal deep learning," in *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pp. 160–167, Rotorua, New Zealand, August 2019.

[21] M. Munir, S. A. Siddiqui, A. Dengel, and S. Ahmed, "Deepant: a deep learning approach for unsupervised anomaly detection in time series," *IEEE Access*, vol. 7, pp. 1991–2005, 2019.

[22] J. Goh, S. Adepu, M. Tan, and S. L. Zi, "Anomaly detection in cyber physical systems using recurrent neural networks," in *2017 IEEE 18th International Symposium on High Assurance Systems Engineering (HASE)*, Singapore, Singapore, January 2017.

[23] Q. Lin, S. Adepu, S. Verwer, and A. Mathur, "Tabor: A graphical model-based approach for anomaly detection in industrial control systems," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security, ASIACCS 18*, New York, NY, USA, May 2018.

[24] J. Inoue, Y. Yamagata, Y. Chen, C. M. Poskitt, and J. Sun, "Anomaly detection for a water treatment system using unsupervised machine learning," in *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 1058–1065, New Orleans, LA, USA, November 2017.

[25] J. Goh, S. Adepu, K. N. Junejo, and A. Mathur, "A dataset to support research in the design of secure water treatment systems," in *Critical Information Infrastructures Security. CRITIS 2016. Lecture Notes in Computer Science, vol 10242*, G. Havarneanu, R. Setola, H. Nassopoulos, and S. Wolthusen, Eds., Springer, Cham, 2016.

[26] D. Shalyga, P. Filonov, and A. Lavrentyev, "Anomaly detection for water treatment system based on neural network with automatic architecture optimization," 2018, http://arxiv.org/abs/1807.07282.

[27] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, San Diego, CA, USA, 2015.