

## Article

# Deep Learning-Based Cyber–Physical Feature Fusion for Anomaly Detection in Industrial Control Systems

Yan Du <sup>1</sup>, Yuanyuan Huang <sup>1,\*</sup>, Guogen Wan <sup>1</sup> and Peilin He <sup>2</sup><sup>1</sup> Department of Network Engineering, Chengdu University of Information Technology, Chengdu 610225, China<sup>2</sup> Department of Informatics and Networked Systems, University of Pittsburgh, Pittsburgh, PA 15260, USA

\* Correspondence: iyyhuang@hotmail.com or hy@cuit.edu.cn

**Abstract:** In this paper, we propose an unsupervised anomaly detection method based on the Autoencoder with Long Short-Term Memory (LSTM-Autoencoder) network and Generative Adversarial Network (GAN) to detect anomalies in industrial control system (ICS) using cyber–physical fusion features. This method improves the recall of anomaly detection and overcomes the challenges of unbalanced datasets and insufficient labeled samples in ICS. As a first step, additional network features are extracted and fused with physical features to create a cyber–physical dataset. Following this, the model is trained using normal data to ensure that it can properly reconstruct the normal data. In the testing phase, samples with unknown labels are used as inputs to the model. The model will output an anomaly score for each sample, and whether a sample is anomalous depends on whether the anomaly score exceeds the threshold. Whether using supervised or unsupervised algorithms, experimentation has shown that (1) cyber–physical fusion features can significantly improve the performance of anomaly detection algorithms; (2) the proposed method outperforms several other unsupervised anomaly detection methods in terms of accuracy, recall, and F1 score; (3) the proposed method can detect the majority of anomalous events with a low false negative rate.

**Citation:** Du, Y.; Huang, Y.; Wan, G.; He, P. Deep Learning-Based Cyber–Physical Feature Fusion for Anomaly Detection in Industrial Control Systems. *Mathematics* **2022**, *10*, 4373. <https://doi.org/10.3390/math10224373>

Academic Editor: Jonathan Blackledge

Received: 23 October 2022

Accepted: 18 November 2022

Published: 20 November 2022

**Publisher’s Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

**Keywords:** deep learning; anomaly detection; cyber–physical; industrial control systems

**MSC:** 68T09

## 1. Introduction

In recent years, cyberattacks have caused significant damage to industrial production and national infrastructure [1]; the Stuxnet virus swept the global industry in 2010 and was able to carry out targeted attacks on infrastructure, with Iran suffering the most severe effects [2]. In 2015, a malicious program called BlackEnergy affected multiple substations in the Ukrainian power sector [3]. Many Ukrainian government agencies and companies were attacked by the ransomware NotPetya in 2017, which ultimately caused havoc worldwide [4]. A serious disaster can also result from the failure of hardware or software within an ICS as well as threats from the Internet. Globally, ICS security incidents occur frequently.

In order to secure ICS, anomaly detection is a promising approach [5]. It is usually physical faults or network attacks that cause anomalous events to occur in ICS. Sensors, actuators, pipelines, and other industrial equipment may malfunction due to physical faults. A network attack refers to an attack on a communication channel, host, or process control system, such as a man-in-the-middle attack (MITM), a denial of service (DoS), or a scanning attack. The purpose of industrial sensors is to collect status information (referred to in this paper as physical information) about the various industrial equipment in the system and to reflect the physical processes that take place within it. Physical faults have an impact on the physical operation of the system, but not on its network traffic. This

results in physical faults not being detected by anomaly detection methods based solely on network traffic. The physical processes of a system may not necessarily be affected by some network attacks. Consequently, algorithms that detect anomalies based solely on physical information are not able to detect these attacks. The use of anomaly detection algorithms that are based solely on physical information cannot detect network attacks in a timely manner, since most network attacks against ICS do not immediately cause the system to enter an abnormal state. Our conclusion is that taking into account both network traffic information and physical information is an effective way to improve the detection performance for anomaly detection algorithms that are used in industrial control systems [6], which has tended to be ignored in past studies.

In the past decade, artificial intelligence (AI) has been rapidly developed and applied in various fields [7–10]. A number of AI-based approaches have emerged in ICS security, which can be categorized as supervised and unsupervised algorithms as a result of the success of AI in traditional IT security [11]. In the past, many anomaly detection algorithms based on supervised algorithms have been proposed. Although the industrial Internet continues to develop, attacks from the Internet are emerging in new ways, and supervised algorithms have a limited ability to detect unknown attacks, making them increasingly unsuitable for ICS security. As ICS datasets have significant imbalances and abnormal data are much smaller than normal data, coupled with a lack of sufficient labeled samples, supervised algorithms are no longer suitable for application in ICS security problems. The limitations of supervised learning can be overcome by unsupervised algorithms such as One-Class SVM (OCSVM) [12] and isolation forests [13].

Autoencoder is an unsupervised algorithm that contains an encoder and a decoder [14]. The input  $X$  is mapped by the encoder to the latent variable  $Z$ , and subsequently  $Z$  is mapped by the decoder to the reconstruction  $R$ . The deviation between the input  $X$  and the reconstruction  $R$  is called reconstruction error. For autoencoder-based anomaly detection, the reconstruction error is used to calculate an anomaly score. Detecting anomalies can be accomplished using autoencoders trained using only normal data. When training a model, it is assumed that the model will only learn how to reconstruct for normal samples. During the testing phase, the model may not be able to reconstruct the anomaly sample well, so the anomaly sample will produce a higher reconstruction error compared to the reconstruction error of the normal sample. In some cases, small anomalies can lead to small reconstruction errors, making it difficult to detect small anomalies. Generative adversarial networks (GANs) may be used to identify small anomalies and amplify reconstruction errors [15]. Autoencoders and GANs are both unsupervised artificial neural networks, with the difference being that GANs include an adversarial game mechanism [16]. The goal of training the generator is to generate data that are as realistic as possible and thus fool the discriminator (i.e., maximizing the likelihood that the discriminator will be incorrect). As well as a generator, the GAN contains a discriminator. When training a discriminator, the objective is to minimize its own error probability, i.e., to be able to distinguish with high accuracy whether the data are real or generated. Due to the time series nature of ICS data, individual samples cannot be considered independently. Compared with ordinary autoencoders, LSTM-based autoencoders [17] have more powerful capability in reconstructing time series data.

In light of the above issues, the main contributions of this paper include the following:

- Based on the latest public ICS dataset, a method for extracting system network features is designed for ICS, and the original physical features are fused with additionally extracted network features to create a cyber–physical dataset with fusion features.
- A model is proposed for unsupervised anomaly detection for ICS based on LSTM-Autoencoder and GAN, which is evaluated using the cyber–physical dataset. In terms of precision, recall, and F1-score, the model outperforms several other methods.

- Both supervised and unsupervised algorithms are used to investigate the effects of additional extracted network features on anomaly detection results. As a result of the experiments performed in this paper, it has been found that the features extracted from the network can significantly improve the performance of the anomaly detection algorithm.

Acquiring data from industrial sensors is an inherent function of ICS, and ICS network traffic data can be collected by listening to communication channels. It is feasible to collect both network data and physical data, and then extract the cyber–physical fusion features. The massive amount of data generated during the normal operation of the ICS is sufficient to train the unsupervised model. Without significantly changing the components of the ICS, the models need to be trained only once to detect anomalies, including various novel attack methods. It is undeniable that the components of an ICS are fixed for a long time, and the network topology is not easily changed. Therefore, the unsupervised anomaly detection model using the cyber–physical fusion features proposed in this paper can help industrial control systems cope with various cyber and physical threats, attacks, and challenges in a cost-effective and profitable manner.

In the remainder of this paper, the following sections are presented: Section 2 discusses related work in the area of anomaly detection of ICS; Section 3 describes the dataset used in this paper; Section 4 describes the method proposed in this paper; Section 5 describes the experimental setup and presents the experimental results and analysis; and Section 6 concludes with our future plans.

## 2. Related Work

It is necessary to detect anomalies in ICS in order to ensure its security. Studies conducted in the past can be categorized according to their use of physical information or network traffic, depending on the features selected.

- (1) Studies using physical information. Industrial sensors collect physical data such as water level, temperature, and humidity. Ahmed et al. [18] use the hardware characteristics of the sensor and the physical characteristics of the process to create a unique fingerprint for each sensor. In normal operation, noise-based fingerprints are created and can be used to detect attacks by comparing the differences between the noise pattern and the fingerprint pattern. According to Lin et al. [19], timed automata can be used to learn the laws that govern the change of sensor value. Furthermore, sensor and actuator dependencies are analyzed using a Bayesian network. The method is capable of detecting anomalies and locating the abnormal sensor or actuator. Industrial sensor data can be analyzed based on their time and frequency characteristics. In their study, Nguyen et al. [20] developed a method for detecting outliers in time-frequency data using continuous wavelet transforms. The authors of Zhao et al. [21] proposed a correlation-based method for detecting anomalies using sensor data and the correlations between them. Compared with only using sensor data, their method achieved higher accuracy.
- (2) Studies using network traffic. Since ICS networks are more stable than IT networks, abnormal network traffic usually indicates that the system is being attacked. Network traffic-based anomaly detection methods can be further divided into packet-based detection, flow-based detection, and session-based detection [22]. To detect abnormal behavior in ICS, Song et al. [23] extracted the behavioral sequence data from Modbus traffic to model the system's normal behavior, and compared the actual behavioral data with the model's predictions. Lee et al. [24] proposed AE-CGAN (autoencoder-conditional GAN) to oversample rare classes on the basis of the GAN model. It is able to achieve more accurate performance metrics in the case of significant imbalance between normal and abnormal traffic. Benaddi et al. [25] used Distributional Reinforcement Learning (DRL) and GAN to help distributional RL-based IDS enhance the detection of minority network attacks and improve the efficiency

and robustness of anomaly detection systems in the Industrial Internet of Things (IIoT). By extracting the temporal characteristics of the original traffic in the SCADA system, Kalech et al. [26] proposed a method for detecting network anomalies based on temporal pattern recognition. In order to detect abnormal behavior, Hidden Markov models and artificial neural networks are used. A multi-level anomaly detection scheme combining LSTMs and Bloom filters was proposed by Feng et al. [27] in order to detect malicious traffic in SCADA datasets. An algorithm for detecting anomalous traffic was proposed by Zhang et al. [28]. A grayscale image was created by converting the ICS traffic feature values into grayscale images, and then the model was trained with the resulting grayscale images, which improved the accuracy of anomaly detection.

According to another perspective, past research can also be divided into supervised and unsupervised research. The use of supervised machine learning has been demonstrated in some studies [29,30] as a means of detecting anomalous events or attacks. In spite of good results, the system was only able to detect known attacks and not unknown or zero-day attacks. ICS datasets are also often imbalanced, i.e., the anomalous samples are much smaller than the normal samples, which limits the performance of the supervised algorithm. Unsupervised or semi-supervised algorithms have been used in other studies to overcome the limitations of supervised algorithms. According to Kravchik et al. [31], their algorithm was able to detect 31 out of 36 network attacks using a one-dimensional CNN-based semi-supervised algorithm. Chang et al. [32] reported that an anomaly detection framework based on k-means and convolutional autoencoders achieved an F1-score of 0.9373 for water storage tank datasets. An autoencoder-based anomaly detection model was proposed by Audibert et al. [33], which used the reconstruction error as the loss function during the training phase and as the anomaly score during the testing phase. An anomaly is determined when the sample's anomaly score exceeds a predetermined threshold. Using an adaptive update strategy based on WGAN-GP, Lu et al. [34] proposed an improved generative adversarial network that produces fake anomaly samples, improving the accuracy of anomaly detection. Li et al.'s [35] GAN-based semi-supervised method, MAD-GAN, utilizes both LSTMs as generators and discriminators to capture the temporal correlation between time series distributions and potential interactions between variables, and it can detect anomalies effectively.

Anomaly detection algorithms are designed based on the selection of appropriate features. In order to detect anomalies in ICS, it is not enough to rely solely on physical information, but it is also necessary to consider network information. However, there are some limitations to the above methods due to the dataset. It was found that the datasets they selected had the following problems: (1) the dataset was not acquired in an ICS environment; (2) the dataset was nonpublic; (3) the dataset was outdated; and (4) the dataset was either restricted to physical process data or to network traffic. The authors in [36] compared the classification performance achieved by the algorithm when only using network features with that achieved by the algorithm when using physical network features, demonstrating that the fusion of physical and network information contributes to improved classification accuracy. The experiment was conducted on four supervised machine learning algorithms, but unsupervised algorithms were not considered.

Due to the above deficiencies, the following improvements have been made in this paper.

- In terms of the dataset, the latest ICS public dataset WDT [37] is utilized, which provides data on physical processes and their corresponding network traffic.
- When extracting features, we take into account the physical information and network traffic of ICS.
- Both supervised and unsupervised algorithms are used in the evaluation of performance to determine whether cyber-physical features contribute to the improvement of anomaly detection.

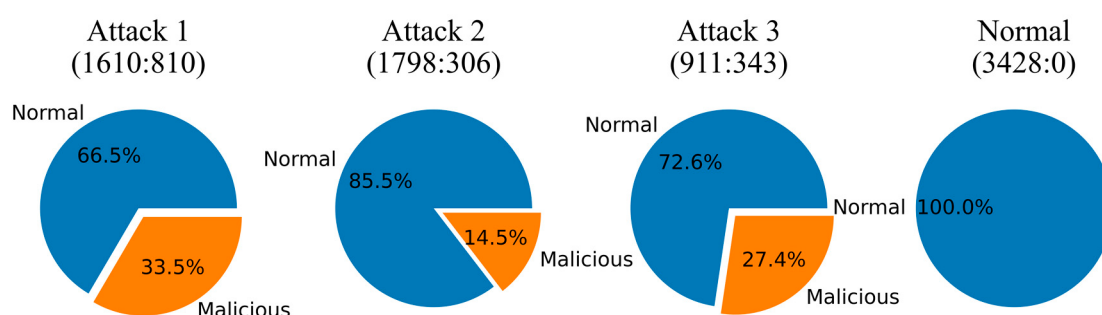
- An unsupervised anomaly detection model based on LSTM-Autoencoder and GAN is proposed, which solves the problem of low recall in past anomaly detection models, and is suitable for the ICS field that does not have sufficient labeled samples.

### 3. Dataset Description

During the normal operation of the Water Distribution Testbed as well as in the event of network attacks or physical faults, the dataset used in this study was compiled from four acquisitions. In Table 1, each acquisition is represented as a sub-dataset. During the first acquisition, eight network attacks or physical failures were conducted, resulting in eight scenarios. In a similar manner, the second and third acquisitions yielded thirteen and seven scenarios, respectively. As of the time of the fourth acquisition, the system was functioning normally, without any network attacks or physical faults. In total, physical faults included two water leaks and six sensors and pumps breakdowns, and network attacks included eight man-in-the-middle (MITM) attacks, five denial of service (DoS) attacks, and seven scanning attacks. Figure 1 shows the number and proportion of samples divided into normal and malicious for each acquisition.

**Table 1.** Data acquisition and description.

| Acquisitions         | Description<br>(Scenario Number: 1.1–1.8, 2.1–2.13, 3.1–3.7)                                                                                                                                                            |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| First<br>(Attack 1)  | phy_att_1.csv, attack_1.pcap<br>5 MITM attack scenarios. (1.1, 1.3, 1.5, 1.7, 1.8)<br>3 physical fault scenarios. (1.2, 1.4, 1.6)                                                                                       |
| Second<br>(Attack 2) | phy_att_2.csv, attack_2.pcap<br>7 scan attack scenarios. (2.1, 2.2, 2.3, 2.4, 2.7, 2.9, 2.11)<br>3 Dos attack scenarios. (2.5, 2.10, 2.13)<br>2 physical fault scenarios. (2.6, 2.8)<br>1 MITM attack scenarios. (2.12) |
| Third<br>(Attack 3)  | phy_att_3.csv, attack_3.pcap<br>3 physical fault scenarios. (3.1, 3.3, 3.4)<br>2 Dos attack scenarios. (3.2, 3.5)<br>2 MITM attack scenarios. (3.6, 3.7)                                                                |
| Fourth<br>(Normal)   | phy_normal.csv, normal.pcap<br>No attack.                                                                                                                                                                               |



**Figure 1.** Number and proportion of samples divided into normal and malicious.

This dataset provides both the physical process data and the corresponding raw network traffic. The physical process data describe the information for the 40 physical statuses of the system in every second, such as whether the pump is turned on and the pressure sensor value of the water tank. In addition, the dataset also provides some network

features extracted from raw network traffic data, such as the IP and MAC addresses of packets. For more detailed information on this dataset, please refer to [37].

#### 4. Methodology

Firstly, we describe how to extract additional network features and fuse them with the original physical features. Then we formulate the problem, and finally we describe our proposed anomaly detection model in more detail.

##### 4.1. Extraction and Fusion of Additional Network Features

The physical information collected by industrial sensors alone is not capable of detecting abnormal behavior in time owing to the widespread adoption of traditional information technology in ICS, and the damage caused by cyberattacks has a hysteresis. It is therefore important to take into account both physical information and network traffic when extracting features for anomaly detection.

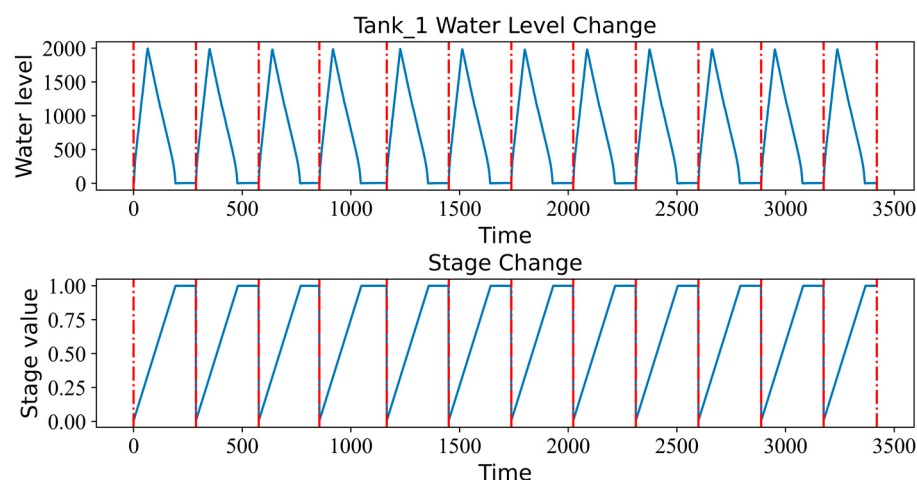
The original physical process datasets have a sampling interval of one second, while the number of samples collected per second in the original network datasets is over 1000. By re-extracting the network features from the original network traffic according to the specifics of the ICS network, we can fuse the physical and network features together and summarize the situation every second. To enhance the performance of anomaly detection, 22 additional features were extracted from the original dataset. In Table 2, you will find a list of the new features that have been added.

**Table 2.** Additional extracted features.

| No. | Features         | Description                                      |
|-----|------------------|--------------------------------------------------|
| 1   | pkt_num          | Number of all types of packets                   |
| 2   | icmp_pkt_num     | Number of ICMP packets                           |
| 3   | arp_pkt_num      | Number of ARP packets                            |
| 4   | tcp_pkt_num      | Number of TCP packets                            |
| 5   | mb_q_num         | Number of MODBUS request packets                 |
| 6   | mb_r_num         | Number of MODBUS response packets                |
| 7   | avg_pkt_size     | Average packet byte size                         |
| 8   | avg_payload_size | Average packet payload byte size                 |
| 9   | mb_q_avg         | Average MODBUS request packet payload byte size  |
| 10  | mb_r_avg         | Average MODBUS response packet payload byte size |
| 11  | illegal_mac      | Illegal MAC address appears                      |
| 12  | illegal_ip       | Illegal IP address appears                       |
| 13  | fc1_pkt_num      | Number of Modbus packets with function code 1    |
| 14  | fc3_pkt_num      | Number of Modbus packets with function code 3    |
| 15  | fc5_pkt_num      | Number of Modbus packets with function code 5    |
| 16  | fc6_pkt_num      | Number of Modbus packets with function code 6    |
| 17  | fin_flag_num     | Number of packets with FIN in the TCP flag       |
| 18  | syn_flag_num     | Number of packets with SYN in the TCP flag       |
| 19  | rst_flag_num     | Number of packets with RST in the TCP flag       |
| 20  | psh_flag_num     | Number of packets with PSH in the TCP flag       |
| 21  | ack_flag_num     | Number of packets with ACK in the TCP flag       |
| 22  | stage            | Stage of the current moment in a process cycle   |

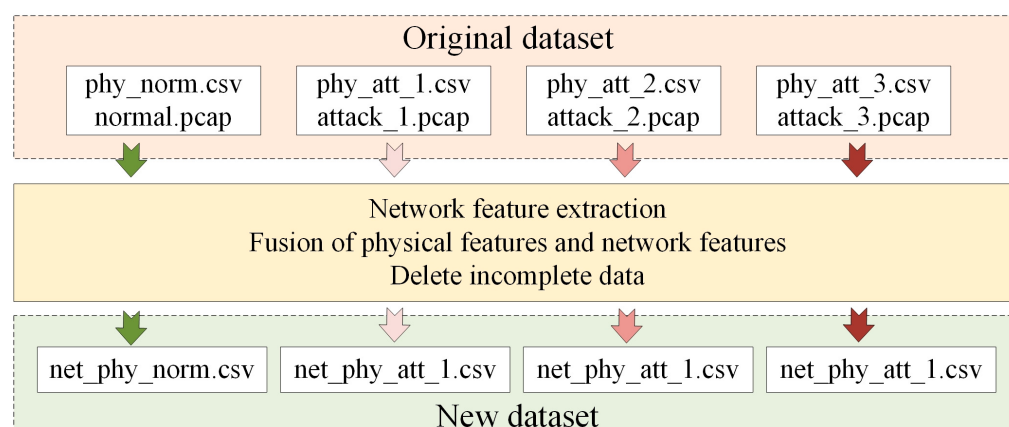
In addition, a feature named stage is added, which describes the stage of the current moment in the process cycle, and its value range is (0,1]. Taking the fourth acquisition as an example, there are a total of 3423 sampling points, including 12 complete process cycles. As shown in Figure 2, in each process cycle, the water level of Tank\_1 gradually increases from 0 to the maximum value, and then gradually decreases to 0 and maintains

for a period of time. Correspondingly, the value of stage is gradually increased from 0 to 1 and maintained for a period of time.



**Figure 2.** Changes of stage value and Tank\_1 water level during process cycle.

The architecture of additional feature extraction and fusion is shown in Figure 3. For each row (sample) in the original physical dataset, its sampling time is time  $t$  (e.g., 09/04/2021 11:30:55). All packets with time  $t$  are aggregated from the network traffic corresponding to this physical dataset, and the features described in Table 2 are extracted from those packets. Subsequently, the newly extracted features are fused with the original physical features to form a cyber-physical dataset. Some incomplete data were deleted, which were mainly concentrated in the first and last part of the dataset. The reason for the incomplete data is that when the original physical dataset is acquired, the corresponding original network dataset has not yet been acquired or the acquisition has been completed. The information of the finally formed cyber-physical dataset is shown in Table 3.



**Figure 3.** Additional network feature extraction and fusion architecture.

**Table 3.** Cyber–physical dataset after feature fusion.

| Dataset           | Number of Samples and Features |
|-------------------|--------------------------------|
| net_phy_att_1.csv | (2409, 62)                     |
| net_phy_att_2.csv | (2092, 62)                     |
| net_phy_att_3.csv | (1248, 62)                     |
| net_phy_norm.csv  | (3421, 62)                     |
| Total samples     | 9170                           |

#### 4.2. Problem Formulation

In this paper, a dataset with sample number  $T$  is considered a multivariate time series  $TS$  of length  $T$ .  $x_t$  is a vector consisting of all physical features and network features at time  $t$ , and the number of features is  $m$ .

$$TS = \{x_1, x_2, \dots, x_T\} (x_t \in \mathbb{R}^m, 1 \leq t \leq T) \quad (1)$$

In order to make better use of the correlation between observations at the current moment and previous observations, a time window  $W_t$  is defined. For each observation, its correlation with the previous  $K$  observations is considered. Therefore, the original time series  $TS$  can be transformed into a time window series  $W$ .

$$W = \{W_1, W_2, \dots, W_T\} (W_t = \{x_{t-K+1}, \dots, x_{t-1}, x_t\} \in \mathbb{R}^{K \times m}, 1 \leq t \leq T) \quad (2)$$

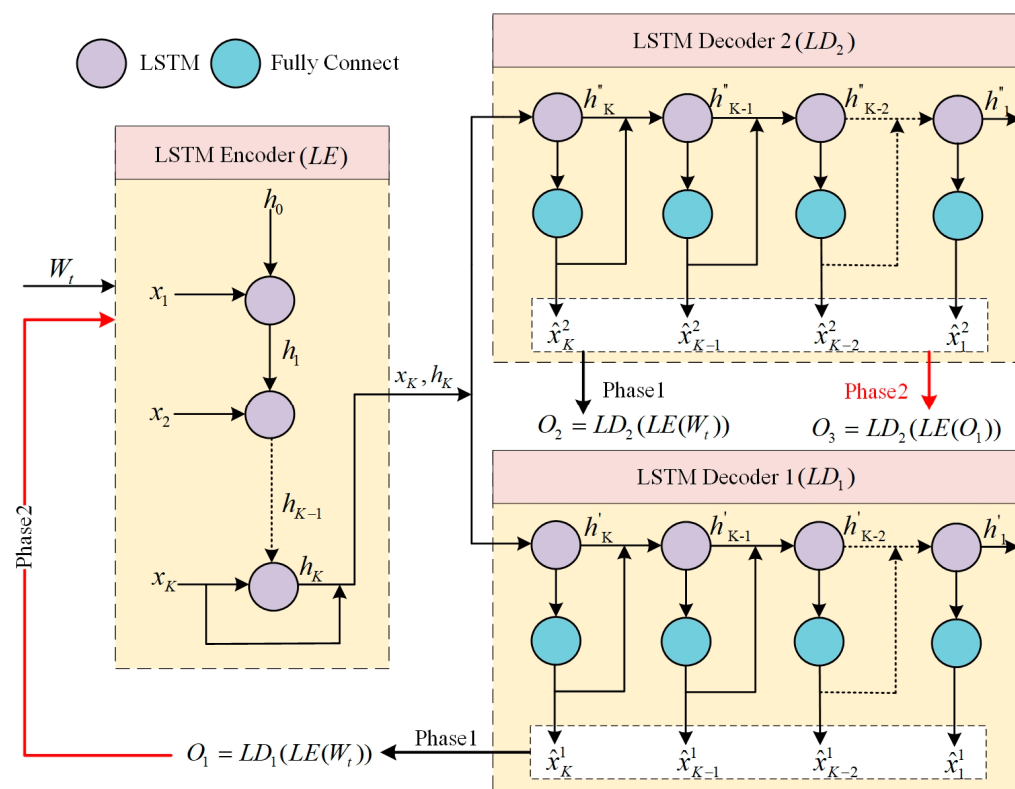
Use the time window series  $W$  as the input to the model instead of the raw time series  $TS$ . Before conversion to a time window series, each observation  $x_t$  in the  $TS$  was normalized by

$$TS^j = \{x_1^j, x_2^j, \dots, x_T^j\} (x_t^j = \frac{x_t^j - \min(TS^j)}{\varepsilon + \max(TS^j) - \min(TS^j)}, 1 \leq j \leq m, 1 \leq t \leq T) \quad (3)$$

where  $\varepsilon$  is a very small number in order to prevent zero-division.

#### 4.3. Proposed Model

The proposed model consists of three modules: an encoder network  $LE$  using LSTM, and two decoder networks  $LD_1$  and  $LD_2$  using LSTM. As can be seen from Figure 4, these three modules constitute two LSTM-Autoencoders  $LAE_1$  and  $LAE_2$  that share the encoder network. The hyperparameters of the model are shown in Table 4. The training of the model consists of two phases.



**Figure 4.** Proposed model architecture. The proposed model consists of three modules: an encoder network  $LE$ , and two decoder networks  $LD_1$  and  $LD_2$ .

**Table 4.** Model hyperparameters.

| Module                          | Hyperparameter                                    | Value     |
|---------------------------------|---------------------------------------------------|-----------|
| LSTM encoder                    | Layer of LSTM                                     | 1         |
|                                 | Input size and hidden size for each layer of LSTM | (62, 128) |
|                                 | Dropout                                           | 0.2       |
| LSTM decoder1 and LSTM decoder2 | Layer of LSTM                                     | 1         |
|                                 | Input size and hidden size for each layer of LSTM | (62, 128) |
|                                 | Dropout                                           | 0.2       |
| LSTM decoder2                   | Layer of Dense                                    | 1         |
|                                 | Size of each layer of the Dense                   | (128, 62) |

#### 4.3.1. Phase 1—Input Reconstruction

The goal of this phase is to train  $LAE_1$  and  $LAE_2$  to reconstruct the input. LSTM-Autoencoder can reconstruct each time window  $W_t = \{x_1, \dots, x_{K-1}, x_K\}$ . The time window  $W_t$  is used as the input of the model, and the encoder network  $LE$  will output the hidden variable  $h_K \in \mathbb{R}^n$  ( $n$  is the number of cells in the LSTM hidden layer). Then, the two decoder networks will output the reconstructions of  $W_t$  ( $O_1$  and  $O_2$ ) according to  $h_K$  and  $x_K$  in reverse order, where  $x_K$  is the last of  $W_t$ . Use L2-norm to define the reconstruction loss for each decoder:

$$O_1 = LAE_1(W_t), O_2 = LAE_2(W_t) \quad (4)$$

$$Loss1 = ||W_t - O_1||_2, Loss2 = ||W_t - O_2||_2 \quad (5)$$

#### 4.3.2. Phase 2—Adversarial Training

In the second phase,  $LAE_1$  and  $LAE_2$  are trained adversarially. Put reconstruction  $O_1$  as input to  $LAE_2$  again and output reconstruction  $O_3$ . The purpose of training  $LAE_2$  is to hope that it can distinguish whether  $O_3$  is the real data or a reconstruction of the output of  $LAE_1$ . Conversely,  $LAE_1$  is trained to fool  $LAE_2$ , that is, making  $LAE_2$  unable to judge whether  $O_3$  is the real data. The training objective is:

$$O_3 = LAE_2(LAE_1(W_t)) \quad (6)$$

$$\min_{LAE_1} \max_{LAE_2} \|W_t - O_3\|_2 \quad (7)$$

Therefore, the goal of  $LAE_1$  is to minimize the distance between  $O_3$  and  $W_t$ , and the goal of  $LAE_2$  is to maximize this distance, and the loss is defined as follows:

$$Loss1 = +\|W_t - O_3\|_2, Loss2 = -\|W_t - O_3\|_2 \quad (8)$$

Then, the evolutionary loss function is used to combine the losses of the two phases as the total loss for each LAE.

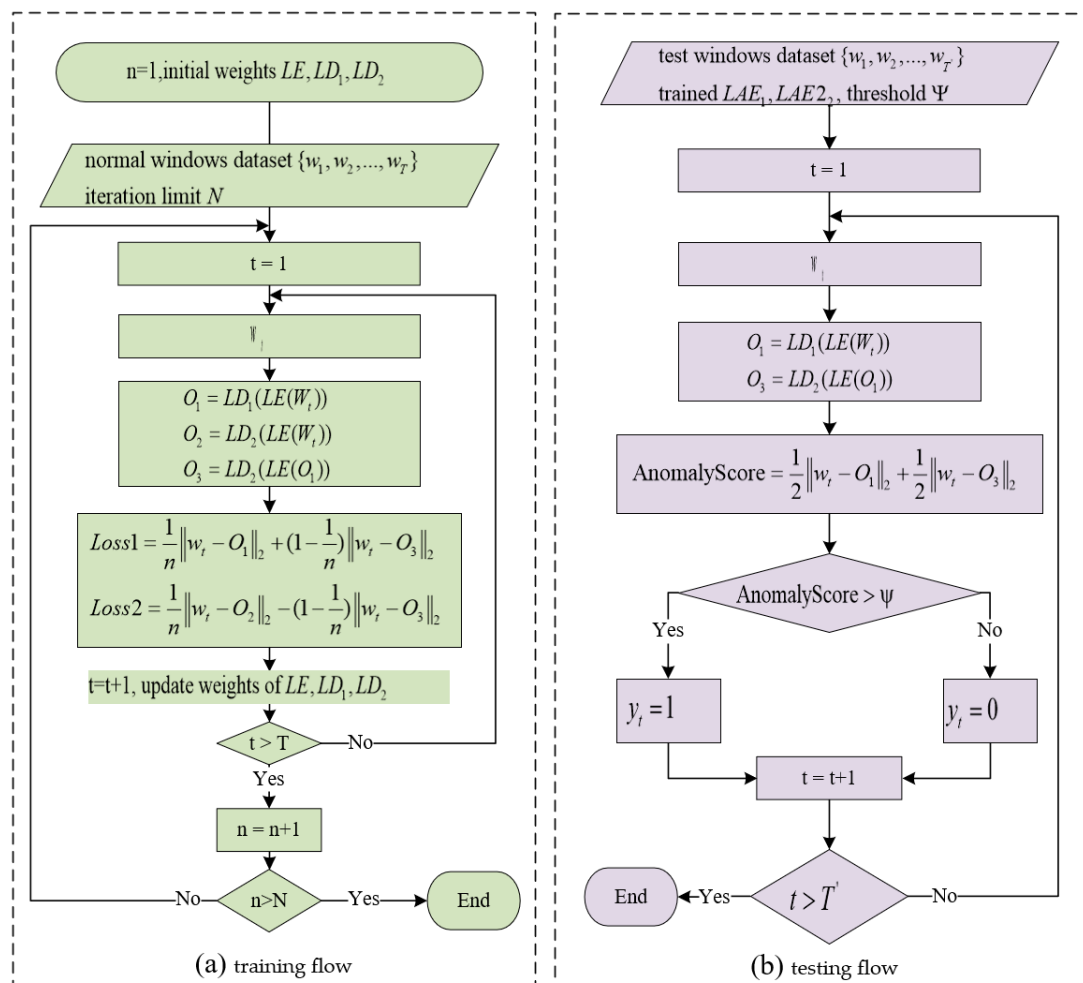
$$Loss1 = \frac{1}{n} \|W_t - O_1\|_2 + (1 - \frac{1}{n}) \|W_t - O_3\|_2 \quad (9)$$

$$Loss2 = \frac{1}{n} \|W_t - O_2\|_2 - (1 - \frac{1}{n}) \|W_t - O_3\|_2 \quad (10)$$

where  $n$  denotes the number of training iterations. The training process of the model can be seen in Figure 5a. Now define the anomaly score:

$$AnomalyScore = \frac{1}{2} \|W_t - O_1\|_2 + \frac{1}{2} \|W_t - O_3\|_2 \quad (11)$$

After the training is completed, the model is used to calculate the anomaly scores for each time window in the normal dataset, and then a threshold is determined based on the distribution of the anomaly scores. During the testing phase, shown in Figure 5b, for each unseen time window, the trained model will output its anomaly score. When the anomaly score of a time window is higher than the threshold, the model judges it as an anomaly.



**Figure 5.** Proposed model training and testing flow chart. (a) Training flow chart; (b) testing flow chart.

## 5. Experiments and Results Analysis

### 5.1. Experiment Environment and Metrics

The experiments were performed using the following hardware and software platforms: Intel(R) Core (TM) i5-12400 CPU, Windows 10 Professional (64 bits), NVIDIA GeForce GTX 1650 Super, NVIDIA CUDA 11.1, Python 3.7.13, Pytorch 1.8.2, Python Scikit-learn library 1.0.2.

The proposed model is evaluated using recall, precision, F1-score, and accuracy.  $TP$ ,  $TN$ ,  $FP$ , and  $FN$  represent true positive, true negative, false positive, and false negative, respectively.

$$Recall = \frac{TP}{TP + FN} \quad (12)$$

$$Precision = \frac{TP}{TP + FP} \quad (13)$$

$$F1 - score = 2 \times \frac{Recall \times Precision}{Recall + Precision} \quad (14)$$

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \quad (15)$$

## 5.2. Dataset

The dataset needs to be divided differently for supervised and unsupervised algorithms.

### 5.2.1. Dataset for Supervised Algorithms

In this paper, the datasets are organized chronologically, with each cyberattack or physical fault lasting for a period of time, corresponding to multiple consecutive samples. When the dataset is shuffled and then divided, some samples from an abnormal event will be placed in the training set, and the remainder will be placed in the testing set. As a result, the model is able to achieve a higher accuracy on the testing set, but this is an illusion [36]. As a result, all samples will be either divided into training sets or testing sets, depending on the scenario. Divide 85% of the normal data into the training set and the rest into the testing set. For the anomaly scenarios, scenario 1.1–1.6, 2.1–2.7, 3.1–3.3 are divided into the training set and the rest are divided into the testing set. Use min–max to normalize the data, and the information of the dataset is shown in Table 5.

**Table 5.** Dataset information for supervised algorithms.

| Label          | Training Set | Testing Set |
|----------------|--------------|-------------|
| Normal         | 5848         | 1864        |
| Physical fault | 426          | 385         |
| MITM           | 358          | 126         |
| DoS            | 67           | 89          |
| Scan           | 5            | 2           |
| Total          | 6704         | 2466        |

### 5.2.2. Dataset for Unsupervised Algorithms

The fourth acquisition (no anomalies) is used as the training set to train the model. The other three acquisitions (with anomalies) were used as testing sets to evaluate the model.

## 5.3. Experiments of Using Supervised Algorithms

Three supervised machine learning algorithms were used for training and testing: random forest (RF), support vector machine (SVM), and naïve Bayes (NB). Use *RandomForestClassifier*, *SVC*, and *GaussianNB* in the Python Scikit-learn library to implement the above algorithm, and the hyperparameters for all of the above algorithms are generated by Python Scikit-learn library 1.0.2 defaulted.

The experimental results are shown in Table 6. All three algorithms achieve poor performance when only using physical features. The best performance is achieved by RF, but its F1 score is only 0.28. When using cyber–physical features, the performance achieved by all three algorithms is greatly improved, with F1 scores exceeding 0.87. The results show that the additionally extracted network features can significantly improve the anomaly detection performance of the supervised algorithm.

**Table 6.** Performance of three supervised machine learning algorithms.

| Algorithm | Physical Features |       |       |       | Cyber–Physical Features |       |       |       |
|-----------|-------------------|-------|-------|-------|-------------------------|-------|-------|-------|
|           | F1                | P     | R     | A     | F1                      | P     | R     | A     |
| RF        | 0.257             | 0.754 | 0.155 | 0.777 | 0.907                   | 1.000 | 0.831 | 0.958 |
| SVM       | 0.126             | 0.764 | 0.068 | 0.763 | 0.895                   | 1.000 | 0.809 | 0.953 |
| NB        | 0.196             | 0.276 | 0.151 | 0.690 | 0.878                   | 0.982 | 0.795 | 0.945 |

## 5.4. Experiments of Unsupervised Algorithms

### 5.4.1. Performance of the Proposed Model

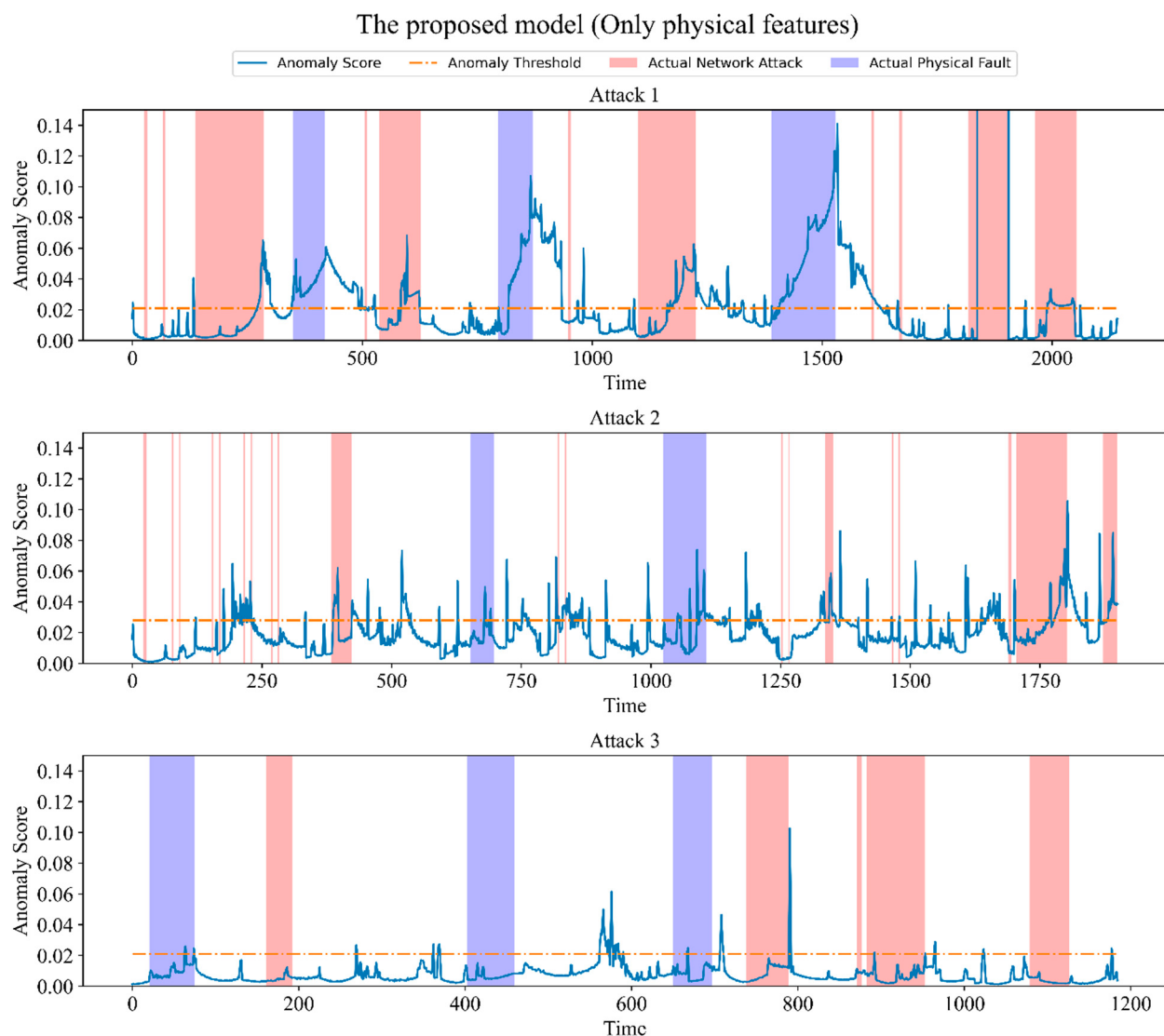
Consider two situations, one using only physical features and another using cyber-physical features. Table 7 shows the performance achieved by the proposed model in the above two situations. In addition, the anomaly scores of the three test sets obtained by the model in the above two situations are shown in Figure 6 and Figure 7, respectively. Cyberattacks and physical faults are marked in red and blue, respectively, in the figure.

When using only physical features, the model performed poorly on all test sets. Conversely, when combining additionally extracted network features, the performance is greatly improved on each test set. We believe that the reason for the poor results obtained by physical features alone is that there are some network attacks that do not affect the physical state of the system too much, so the model fails to detect these network attacks. For the network attack scenario, the anomaly score given by the model for anomalous time points is significantly higher than that for non-anomalous time points, indicating that the model can easily detect network attack events. For physical fault scenarios, the anomaly scores given to anomalous time points are not very significant, but are sufficient to detect most physical fault events.

**Table 7.** Performance of the proposed model.

| Acquisition | Physical Features |       |       |       | Cyber-Physical Features |              |              |              |
|-------------|-------------------|-------|-------|-------|-------------------------|--------------|--------------|--------------|
|             | F1                | P     | R     | A     | F1                      | P            | R            | A            |
| Attack 1    | 0.574             | 0.574 | 0.574 | 0.657 | 0.827                   | 0.827        | 0.826        | 0.860        |
| Attack 2    | 0.292             | 0.293 | 0.292 | 0.730 | 0.646                   | 0.646        | 0.645        | 0.865        |
| Attack 3    | 0.029             | 0.143 | 0.016 | 0.667 | 0.692                   | 0.957        | 0.542        | 0.851        |
| Sum         | 0.425             | 0.479 | 0.382 | 0.686 | <b>0.758</b>            | <b>0.800</b> | <b>0.720</b> | <b>0.860</b> |

Due to the continuous increase in the degree of impact of an attack or fault on the system, it may not cause immediate damage to the system at the beginning, resulting in false negatives. Furthermore, it may still take some time for the attacked system to return to normal after the attack has ended, which may result in false positives. An example would be Scenario 1.6, which simulates the rise of the water level in Tank 3 as a result of a leak in the pipeline. A graph of the water level in Tank 3 over time is shown in Figure 8a. Figure 8b shows the corresponding anomaly scores, as well as the time period during which the fault occurred (scenario 1.6). While the water level rose initially, it was consistent with the normal rise in the tank's level. In this period, the anomaly score does not exceed the threshold, and the model considers it to be a normal period. Persistent faults cause the water level to exceed the normal level and continue to rise. As a result, the anomaly score for this period gradually increases and exceeds the threshold. Upon the resolution of the fault, the water level begins to decline, which is reflected in the anomaly score as well. Nevertheless, the water level remains above the normal level for a period of time after the faults have been resolved, so the anomaly score remains above the threshold, and the model still considers the system to be abnormal.

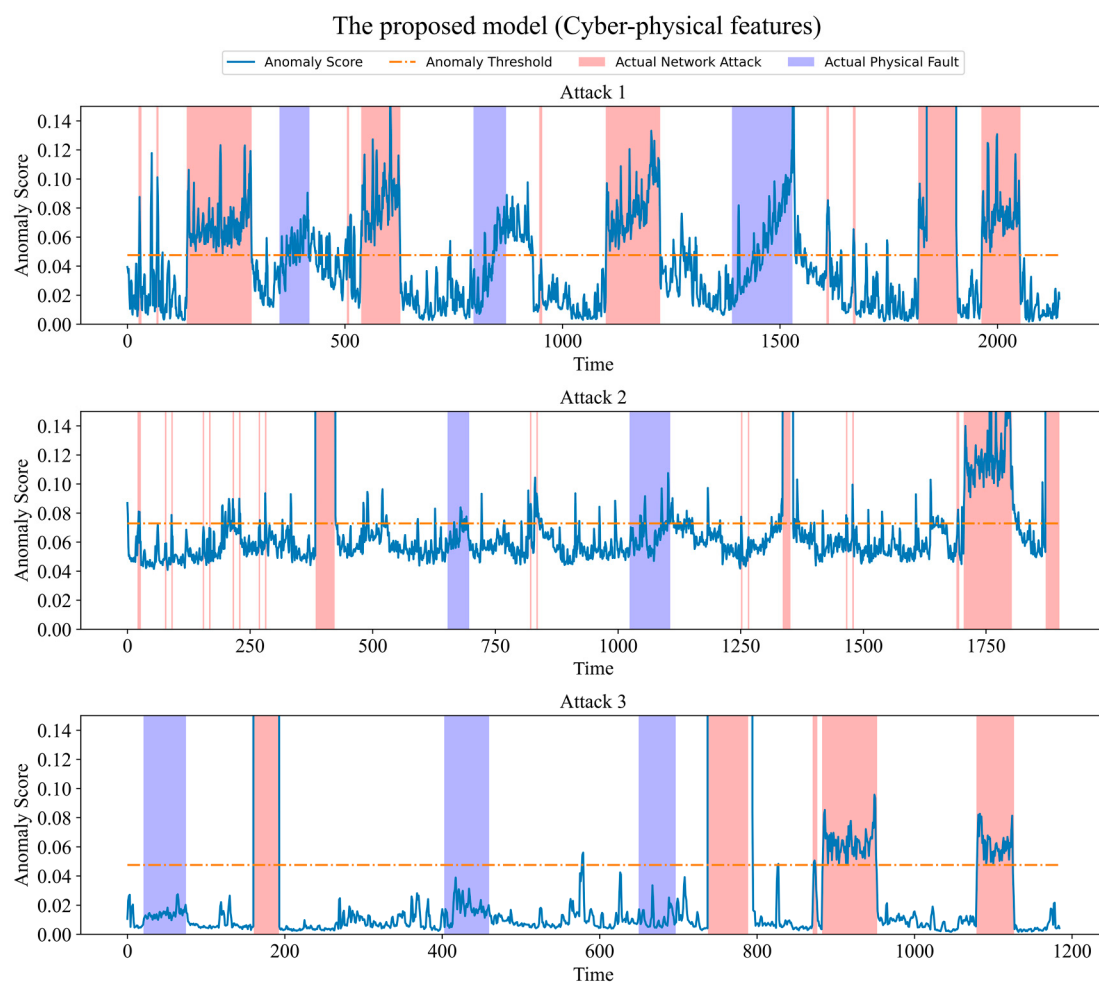


**Figure 6.** Anomaly scores for the proposed model (using only physical features).

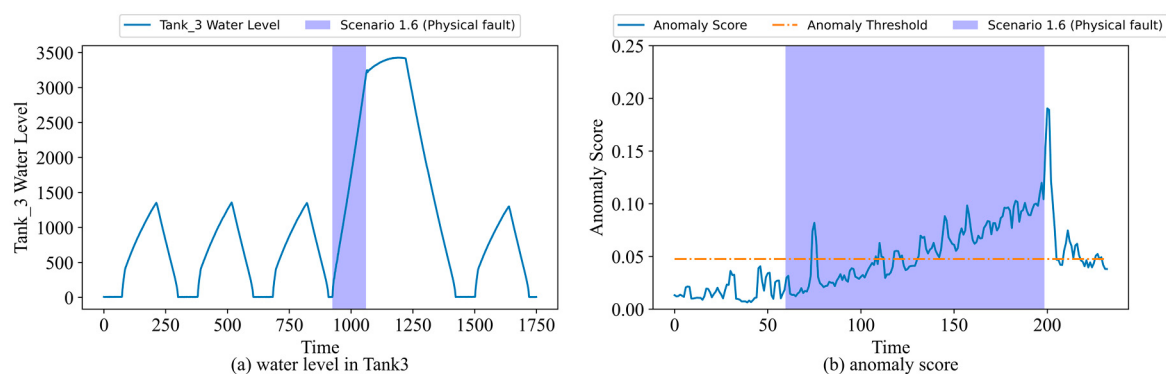
Figure 9a shows the changes of the water levels of Tank 1 and Tank 5 over time, and Figure 9b shows the corresponding anomaly scores. The time periods of the three fault scenarios are marked by red, green, and blue, respectively. Scenario 3.1 simulates a fault that pauses the transfer of water from Tank 1 to Tank 5. Scenario 3.3 simulates a fault by closing the Tank 5 outlet valve, thus achieving a slowdown in the flow of water from Tank 5. Scenario 3.4 simulates a fault that suspends the transfer of water from the reservoir to Tank 1. None of the above three faults caused the water level to exceed the normal level, so none of the anomaly scores exceeded the threshold and the model considered the system to be in a normal state.

#### 5.4.2. Comparison with Other Unsupervised Algorithms

This section compares the performance of OCSVM [12], Isolation Forest (iForest) [13], USAD [33], and the proposed model. This paper implements USAD based on the author's GitHub repository. Both One-Class SVM and Isolation Forest are provided by the Python Scikit-learn library and use default parameters.



**Figure 7.** Anomaly scores for the proposed model (using cyber-physical features).

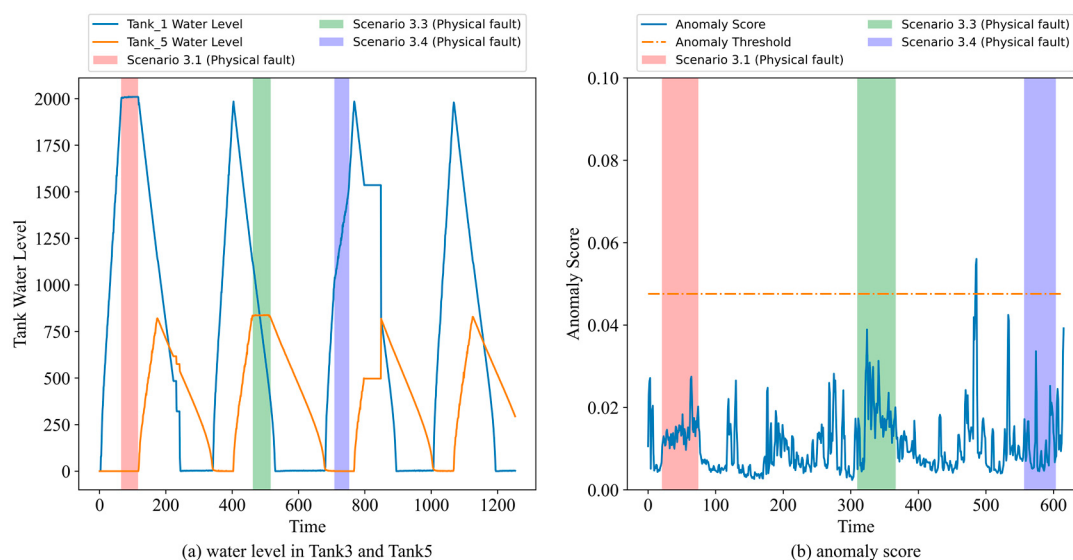


**Figure 8.** Scenario 1.6. (a) Water level in Tank 3; (b) anomaly score.

As can be seen in Table 8, the proposed model outperforms several other algorithms. OCSVM and iForest achieved a high recall rate, but too many false positives resulted in a low F1 score. Compared with the first two algorithms, the F1 score of USAD has been greatly improved, but the recall rate is lower. Low recall means that there are more false negatives, meaning that the model does not effectively detect anomalies, which is fatal for anomaly detection systems. As shown in Figure 10, USAD is able to detect most network attacks, but it is almost incapable of detecting physical faults. In contrast, the proposed model can detect most physical faults. The experimental results show that for the ICS anomaly detection task, the model proposed in this paper can achieve better performance.

**Table 8.** Performance comparison of the proposed model with other methods.

| Methods        | F1    | P     | R     | A     |
|----------------|-------|-------|-------|-------|
| OC-SVM         | 0.489 | 0.324 | 0.999 | 0.364 |
| iForest        | 0.484 | 0.341 | 0.834 | 0.458 |
| USAD           | 0.622 | 0.632 | 0.613 | 0.774 |
| Proposed model | 0.758 | 0.800 | 0.720 | 0.860 |

**Figure 9.** Scenario 3.1, 3.3, 3.4. (a) Water level in Tank 3 and Tank 5; (b) anomaly score.

### 5.5. Ablation Experiments

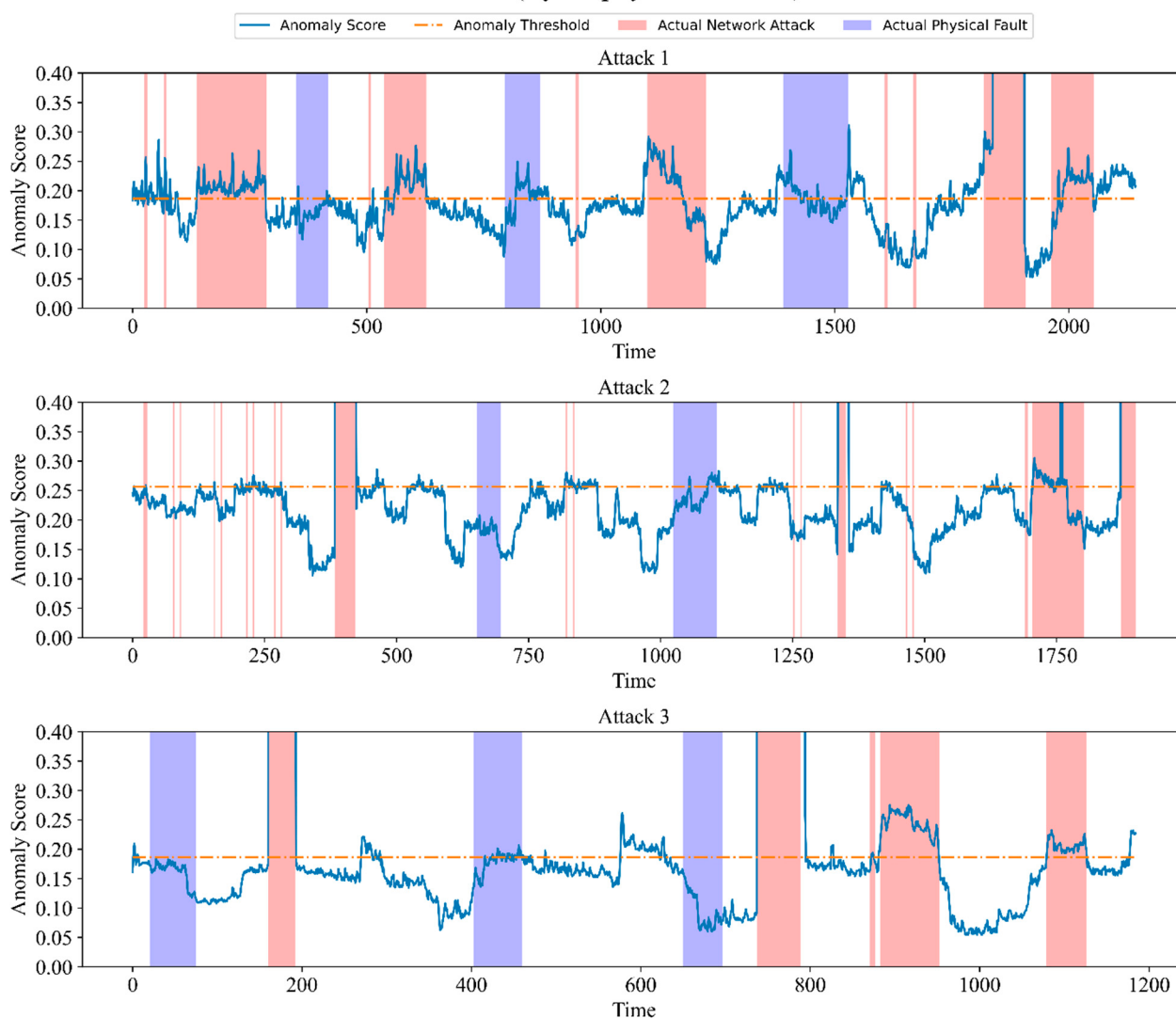
The LSTM autoencoder in the proposed model is replaced by the standard autoencoder, BiLSTM autoencoder, and GRU autoencoder, and their hyperparameters are shown in Table 9. We removed the adversarial training phase from the proposed model, which is hereafter referred to as the proposed model with no adversarial training. The same training settings were set for the above models: the batch size is 32, the window size is 3, the optimizer is Adam, the learning rate is 0.001, the max epoch is 100, and the initial parameters are generated by Pytorch-1.8.2 defaulted.

**Table 9.** Three autoencoder hyperparameters.

| Category             | Hyperparameter  | Value                                               |
|----------------------|-----------------|-----------------------------------------------------|
| Standard autoencoder | Layer of Dense  | 2                                                   |
|                      | Encoder         | Size of the Dense (W means window size)             |
|                      |                 | (62×W, 128)                                         |
|                      | Decoder         | (128, 64)                                           |
|                      |                 | Dropout, Activation function                        |
|                      |                 | 0.1, ReLu                                           |
| BiLSTM autoencoder   | Layer of Dense  | 2                                                   |
|                      | Encoder         | Size of the Dense (W means window size)             |
|                      |                 | (64, 128)                                           |
|                      | Decoder         | (128, 62×W)                                         |
|                      |                 | Dropout, Activation function                        |
|                      |                 | 0.1, ReLu                                           |
| BiLSTM autoencoder   | Layer of BiLSTM | 1                                                   |
|                      | Encoder         | Input size and hidden size for each layer of BiLSTM |
|                      |                 | (62, 128)                                           |
|                      | Dropout         | 0.2                                                 |
|                      | Layer of BiLSTM | 1                                                   |

|                    |         |                                                     |           |
|--------------------|---------|-----------------------------------------------------|-----------|
| GRU<br>autoencoder | Decoder | Input size and hidden size for each layer of BiLSTM | (62, 128) |
|                    |         | Dropout                                             | 0.2       |
|                    | Encoder | Layer of Dense                                      | 1         |
|                    |         | Size of each layer of the Dense                     | (128, 62) |
| GRU<br>autoencoder | Encoder | Layer of GRU                                        | 1         |
|                    |         | Input size and hidden size for each layer of GRU    | (62, 128) |
|                    | Decoder | Dropout                                             | 0.2       |
|                    |         | Layer of GRU                                        | 1         |
| GRU<br>autoencoder | Encoder | Input size and hidden size for each layer of GRU    | (62, 128) |
|                    |         | Dropout                                             | 0.2       |
|                    | Decoder | Layer of Dense                                      | 1         |
|                    |         | Size of each layer of the Dense                     | (128, 62) |

### USAD (Cyber-physical features)



**Figure 10.** Anomaly scores for USAD (using network and physical features).

### 5.6. Discussion

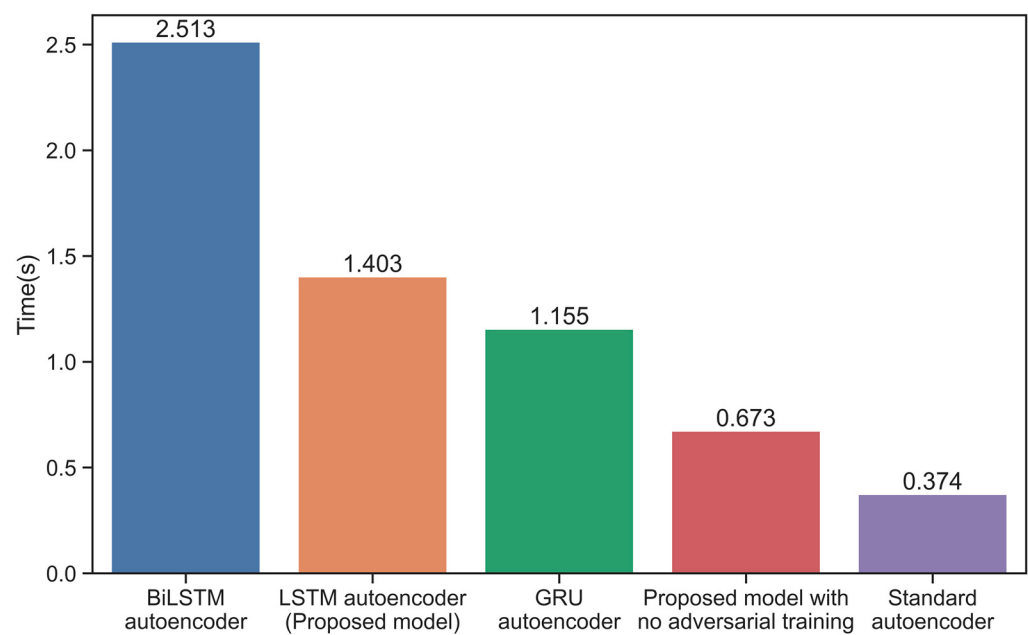
As shown in Table 6, with the addition of network features, the accuracies of RF, SVM, and NB improved from 0.777, 0.763, and 0.690 to 0.958, 0.953, and 0.945, respectively, and the F1 score, precision, recall, and accuracies of the proposed model improved from 0.425, 0.479, 0.382, and 0.686 to 0.758, 0.800, 0.720, and 0.860, respectively. This is due to the existence of some network attacks, such as scanning attacks, which only generate some anomalous network traffic data, but do not have a substantial impact on the physical conduct of the system. Therefore, the fusion of network traffic data and physical sensor data definitely helps to improve the anomaly detection capability.

Compared with other unsupervised algorithms, the unsupervised anomaly detection model proposed in this paper has better performance. As shown in Table 8, the USAD model achieves a recall of only 0.613 when using cyber–physical fusion features, while the proposed model can improve the recall to 0.720, with a performance improvement of about 17.5%. As can be seen from Figures 7 and 10, the USAD model gives anomaly scores for normal and abnormal data that are not very different in general, which means that it does not reconstruct normal data perfectly and therefore cannot clearly distinguish between normal and abnormal samples. In contrast, the proposed model gives a large difference in the abnormal scores for normal and abnormal data, which indicates that the model can detect abnormalities well.

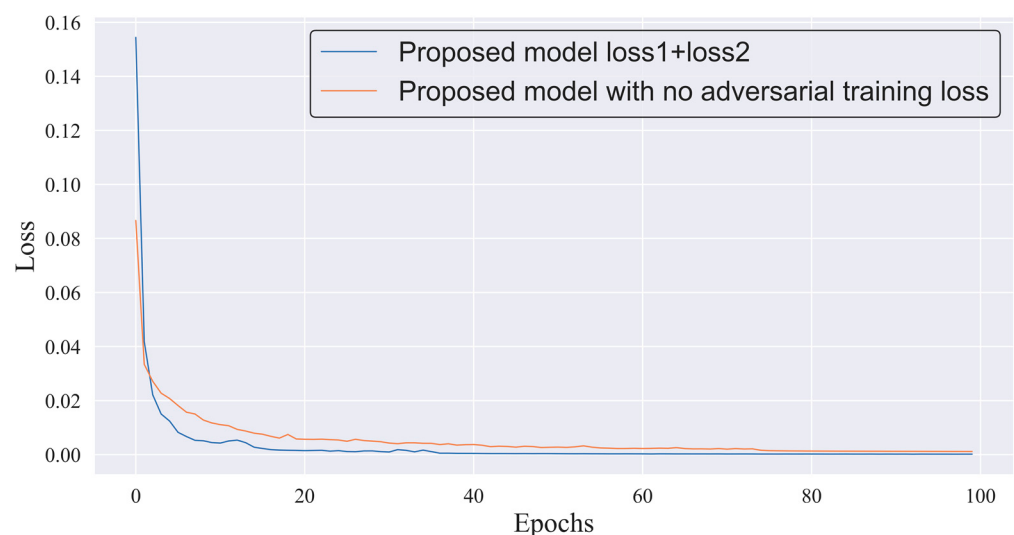
Table 10 depicts the performance of the standard autoencoder, BiLSTM autoencoder, GRU autoencoder, the proposed model (LSTM autoencoder), and the proposed model with no adversarial training, and Figure 11 shows the time they need to consume for one training. It can be seen that the standard autoencoder achieves the fastest training speed as well as the highest recall rate, but its precision and F1 scores are the lowest. This means that the model identifies numerous normal data as abnormal. The BiLSTM autoencoder took more time to train, but the improvement in performance was marginal. The time cost of training the GRU autoencoder is slightly lower than the time cost of training the proposed model, but the performance of the GRU autoencoder is much worse than the proposed model. It achieves a recall of 0.618, while the proposed model achieves a recall of 0.72, which we believe is worth the small time cost to obtain such a significant improvement. As shown in Figure 12, the model is able to reduce the loss earlier and with smaller loss values when adversarial training is performed. Furthermore, when adversarial training is removed, the recall decreases from 0.72 to 0.652, which is sufficient to demonstrate that adversarial training based on generative adversarial networks is indeed able to identify small anomalies by amplifying the reconstruction error.

**Table 10.** Performance comparison of LSTM autoencoder with others.

| Category                                       | F1    | P     | R     | A     |
|------------------------------------------------|-------|-------|-------|-------|
| Standard autoencoder                           | 0.568 | 0.414 | 0.903 | 0.581 |
| BiLSTM autoencoder                             | 0.767 | 0.843 | 0.703 | 0.870 |
| GRU autoencoder                                | 0.729 | 0.888 | 0.618 | 0.860 |
| LSTM autoencoder<br>(proposed model)           | 0.758 | 0.800 | 0.720 | 0.860 |
| Proposed model with no<br>adversarial training | 0.730 | 0.828 | 0.652 | 0.853 |



**Figure 11.** Training time for proposed model and other models.



**Figure 12.** Loss with adversarial training and without adversarial training.

## 6. Conclusions

Given the special characteristics of ICS networks, we designed a method to extract network features. Based on the latest publicly available ICS dataset, the network features are extracted using the previously mentioned method, and then an ICS cyber–physical dataset is created. The anomaly detection algorithm obtained by training with this fused feature has better performance. In addition, we propose an unsupervised anomaly detection method based on LSTM-Autoencoder and GAN. The results of the ablation experiments show that using LSTM as an autoencoder is the optimal choice, and adversarial training based on GAN can also help the model to detect more anomalies.

This paper uses a dataset acquired in ICS using only the Modbus TCP protocol, but other protocols such as S7 and EtherNet/IP exist in the global industry. Our future work will investigate a more effective and compatible method for detecting ICS anomalies based on a more comprehensive dataset.

**Author Contributions:** Conceptualization, Y.D. and Y.H.; funding acquisition, Y.H. and G.W.; methodology, Y.D. and Y.H.; resources, G.W.; software, Y.D.; validation, Y.H., G.W., and P.H.; writing—original draft, Y.D.; writing—review and editing, G.W. and P.H. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was supported by Natural Science Foundation of Sichuan Province (No. 2022NSFSC0557), Foundation of Sichuan Network Culture Research Center (No. WLWH22-18), National Natural Science Foundation of China (No. 62102049, No. 62076042), and Key Research and Development Project of Sichuan (No. 2021YFSY0012, No. 2021YFG0332).

**Data Availability Statement:** The data presented in this study are openly available in the IEEE DataPort at [10.21227/rbvf-2h90] accessed on 20 November 2022, reference number [37].

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Siniosoglou, I.; Radoglou-Grammatikis, P.; Efstathopoulos, G.; Fouliras, P.; Sarigiannidis, P. A Unified Deep Learning Anomaly Detection and Classification Approach for Smart Grid Environments. *IEEE Trans. Netw. Serv. Manag.* **2021**, *18*, 1137–1151. <https://doi.org/10.1109/TNSM.2021.3078381>.
2. Liu, J.; Lin, X.; Chen, X.; Wen, H.; Li, H.; Hu, Y.; Sun, J.; Shi, Z.; Sun, L. ShadowPLCs: A Novel Scheme for Remote Detection of Industrial Process Control Attacks. *IEEE Trans. Dependable Secur. Comput.* **2022**, *19*, 2054–2069. <https://doi.org/10.1109/TDSC.2020.3046267>.
3. Khan, R.; Maynard, P.; McLaughlin, K.; Laverty, D. M.; Sezer, S. Threat Analysis of BlackEnergy Malware for Synchrophasor based Real-time Control and Monitoring in Smart Grid. In Proceedings of the 4th International Symposium for ICS & SCADA Cyber Security Research, Swindon, UK, 23–25 August 2016; pp. 53–63. <https://doi.org/10.14236/ewic/ICS2016.7>.
4. Alladi, T.; Chamola, V.; Zeadally, S. Industrial Control Systems: Cyberattack trends and countermeasures. *Comput. Commun.* **2020**, *155*, 1–8. <https://doi.org/10.1016/j.comcom.2020.03.007>.
5. Fahim, M.; Sillitti, A. Anomaly Detection, Analysis and Prediction Techniques in IoT Environment: A Systematic Literature Review. *IEEE Access* **2019**, *7*, 81664–81681. <https://doi.org/10.1109/ACCESS.2019.2921912>.
6. Ayodeji, A.; Liu, Y.; Chao, N.; Yang, L. A new perspective towards the development of robust data-driven intrusion detection for industrial control systems. *Nucl. Eng. Technol.* **2020**, *52*, 2687–2698. <https://doi.org/10.1016/j.net.2020.05.012>.
7. Zhang, M.; Qu, H.; Belatreche, A.; Chen, Y.; Yi, Z. A highly effective and robust membrane potential-driven supervised learning method for spiking neurons. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *30*, 123–137. <https://doi.org/10.1109/TNNLS.2018.2833077>.
8. Zhang, M.; Wang, J.; Wu, J.; Belatreche, A.; Amornpaisannon, B.; Zhang, Z.; Miriyala V.; Qu H.; Chua Y.; Carlson T.; Li H. Rectified linear postsynaptic potential function for backpropagation in deep spiking neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *33*, 1947–1958. <https://doi.org/10.1109/TNNLS.2021.3110991>.
9. Huang, Y.; Wang, D.; Sun, Y.; Hang, B. A fast intra coding algorithm for HEVC by jointly utilizing naive Bayesian and SVM. *Multimed. Tools Appl.* **2020**, *79*, 33957–33971. <https://doi.org/10.1007/s11042-020-08882-x>.
10. Gou, J.; Sun, L.; Yu, B.; Wan, S.; Ou, W.; Yi, Z. Multi-Level Attention-Based Sample Correlations for Knowledge Distillation. *IEEE Trans. Ind. Inform.* **2022**, 1–11 (early access). <https://doi.org/10.1109/TII.2022.3209672>.
11. Huang, Y.; Lu, J.; Tang, H.; Liu, X. A Hybrid Association Rule-Based Method to Detect and Classify Botnets. *Secur. Commun. Netw.* **2021**, *2021*, 1028878. <https://doi.org/10.1155/2021/1028878>.
12. Amer, M.; Goldstein, M.; Abdennadher, S. Enhancing one-class support vector machines for unsupervised anomaly detection. In Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description, Chicago, IL, USA, 11 August 2013; pp. 8–15. <https://doi.org/10.1145/2500853.2500857>.
13. Liu, F. T.; Ting, K. M.; Zhou, Z. Isolation Forest. In Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, Washington, DC, USA, 15–19 December 2008; pp. 413–422. <https://doi.org/10.1109/ICDM.2008.17>.
14. Bank, D.; Koenigstein, N.; Giryas, R. Autoencoders. *arXiv* **2020**, arXiv:2003.05991.
15. Tuli, S.; Casale, G.; Jennings, N.R. TranAD: Deep Transformer Networks for Anomaly Detection in Multivariate Time Series Data. *arXiv* **2022**, arXiv: 2201.07284.
16. Cai, Z.; Xiong, Z.; Xu, H.; Wang, P.; Li, W.; Pan, Y. Generative Adversarial Networks. *ACM Comput. Surv.* **2021**, *54*, 1–38. <https://doi.org/10.1145/3459992>.
17. Provotar, O.I.; Linder, Y.M.; Veres, M.M. Unsupervised Anomaly Detection in Time Series Using LSTM-Based Autoencoders. In Proceedings of the 2019 IEEE International Conference on Advanced Trends in Information Theory (ATIT), Kyiv, Ukraine, 18–20 December 2019; pp. 513–517. <https://doi.org/10.1109/ATIT49449.2019.9030505>.
18. Ahmed, C.M.; Zhou, J.; Mathur, A.P. Noise matters: Using sensor and process noise fingerprint to detect stealthy cyber attacks and authenticate sensors in cps. In Proceedings of the 34th Annual Computer Security Applications Conference, San Juan PR, USA, 3–7 December 2018; pp. 566–581. <https://doi.org/10.1145/3274694.3274748>.
19. Lin, Q.; Adepu, S.; Verwer, S.; Mathur, A. 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, Incheon, Republic of Korea, 4 June 2018; pp. 525–536. <https://doi.org/10.1145/3196494.3196546>.

20. Nguyen, L.V.; Kapinski, J.; Jin, X.; Deshmukh, J.; Butts, K.; Johnson, T.T. Abnormal Data Classification Using Time-Frequency Temporal Logic. In Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control, Pittsburgh, PA, USA, 18–20 April 2017; pp. 237–242. <https://doi.org/10.1145/3049797.3049809>.
21. Zhao, P.; Kurihara, M.; Tanaka, J.; Noda, T.; Chikuma, S.; Suzuki, T. Advanced correlation-based anomaly detection method for predictive maintenance. In Proceedings of the 2017 IEEE International Conference on Prognostics and Health Management (ICPHM), Dallas, TX, USA, 19–21 June 2017; pp. 78–83. <https://doi.org/10.1109/ICPHM.2017.7998309>.
22. Liu, H.; Lang, B. Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey. *Appl. Sci.* **2019**, *9*, 4396. <https://doi.org/10.3390/app9204396>.
23. Zhanwei, S.; Zenghui, L. Abnormal detection method of industrial control system based on behavior model. *Comput. Secur.* **2019**, *84*, 166–178. <https://doi.org/10.1016/j.cose.2019.03.009>.
24. Lee, J.; Park, K. AE-CGAN Model based High Performance Network Intrusion Detection System. *Appl. Sci.* **2019**, *9*, 4221. <https://doi.org/10.3390/app9204221>.
25. Benaddi, H.; Jouhari, M.; Ibrahim, K.; Ben Othman, J.; Amhoud, E.M. Anomaly Detection in Industrial IoT Using Distributional Reinforcement Learning and Generative Adversarial Networks. *Sensors* **2022**, *22*, 8085. <https://doi.org/10.3390/s22218085>.
26. Kalech, M. Cyber-attack detection in SCADA systems using temporal pattern recognition techniques. *Comput. Secur.* **2019**, *84*, 225–238. <https://doi.org/10.1016/j.cose.2019.03.007>.
27. Feng, C.; Li, T.; Chana, D. Multi-level Anomaly Detection in Industrial Control Systems via Package Signatures and LSTM Networks. In Proceedings of the 2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Denver, CO, USA, 26–29 June 2017; pp. 261–272. <https://doi.org/10.1109/DSN.2017.34>.
28. Zhang, Y.; Li, X.; Li, D.; Yang, H. Abnormal flow monitoring of industrial control network based on convolutional neural network. *J. Comput. Appl.* **2019**, *39*, 1512. <https://doi.org/10.11772/j.issn.1001-9081.2018091928>.
29. Beaver, J.M.; Borges-Hink, R.C.; Buckner, M.A. An Evaluation of Machine Learning Methods to Detect Malicious SCADA Communications. In Proceedings of the 2013 12th International Conference on Machine Learning and Applications, Miami, FL, USA, 4–7 December 2013; pp. 54–59. <https://doi.org/10.1109/ICMLA.2013.105>.
30. Borges Hink, R.C.; Beaver, J. M.; Buckner, M. A.; Morris, T.; Adhikari, U.; Pan, S. Machine learning for power system disturbance and cyber-attack discrimination. In Proceedings of the 2014 7th International Symposium on Resilient Control Systems (ISRCs), Denver, CO, USA, 19–21 August 2013; pp. 1–8. <https://doi.org/10.1109/ISRCs.2014.6900095>.
31. Kravchik, M.; Shabtai, A. Detecting Cyber Attacks in Industrial Control Systems Using Convolutional Neural Networks. In Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and Privacy, Toronto, ON, Canada, 19 October 2018; pp. 72–83. <https://doi.org/10.1145/3264888.3264896>.
32. Chang, C.P.; Hsu, W.C.; Liao, I.E. Anomaly Detection for Industrial Control Systems Using K-Means and Convolutional Auto-encoder. In Proceedings of the 2019 International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Split, Croatia, 19–21 September 2019; pp. 1–6. <https://doi.org/10.23919/SOFTCOM.2019.8903886>.
33. Audibert, J.; Michiardi, P.; Guyard, F.; Marti, S.; Zuluaga, M.A. USAD: UnSupervised Anomaly Detection on Multivariate Time Series. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Virtual Event, USA, 6–10 July 2020; pp. 3395–3404. <https://doi.org/10.1145/3394486.3403392>.
34. Lu, H.; Du, M.; Qian, K.; He, X.; Wang, K. GAN-Based Data Augmentation Strategy for Sensor Anomaly Detection in Industrial Robots. *IEEE Sens. J.* **2022**, *22*, 17464–17474. <https://doi.org/10.1109/JSEN.2021.3069452>.
35. Li, D.; Chen, D.; Jin, B.; Shi, L.; Goh, J.; Ng, S.K. MAD-GAN: Multivariate Anomaly Detection for Time Series Data with Generative Adversarial Networks. In Proceedings of the 28th International conference on artificial neural networks, Munich, Germany, 17–19 September 2020; pp. 703–716. [https://doi.org/10.1007/978-3-030-30490-4\\_56](https://doi.org/10.1007/978-3-030-30490-4_56).
36. Müller, N.; Ziras, C.; Heussen, K. Assessment of Cyber-Physical Intrusion Detection and Classification for Industrial Control Systems. *arXiv* **2022**, arXiv:2202.09352.
37. Faramondi, L.; Flammini, F.; Guarino, S.; Setola, R. A Hardware-in-the-Loop Water Distribution Testbed Dataset for Cyber-Physical Security Testing. *IEEE Access* **2021**, *9*, 122385–122396. <https://doi.org/10.1109/ACCESS.2021.3109465>.