IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 28, 2022, accepted April 8, 2022, date of publication April 18, 2022, date of current version April 25, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3167814

Industrial Control System Anomaly Detection and Classification Based on Network Traffic

JEHN-RUEY JIANG, (Member, IEEE), AND YAN-TING CHEN
Department of Computer Science and Information Engineering, National Central University, Taoyuan City 320317, Taiwan
Corresponding author: Jehn-Ruey Jiang (jrjiang@csie.ncu.edu.tw)

This work was supported in part by the Ministry of Science and Technology (MOST), Taiwan, under Grant 109-2622-E-008-028.

ABSTRACT This paper proposes an anomaly detection and classification method for industrial control systems (ICSs). The proposed method is based on network traffic data of industrial field protocols like Modbus TCP and S7 Communication. First, the denoising autoencoder (DAE) is utilized to reduce data noise and extract core features from data. Second, the synthetic minority oversampling technique (SMOTE) and the Tomek link (T-Link) mechanism are employed to oversample and undersample data for addressing the data imbalance problem. Finally, extreme gradient boosting (XGBoost) is used to leverage the ensemble learning concept to avoid overfitting for achieving good performance. A real-life railway industry ICS dataset called Electra is used to evaluate the performance of the proposed method, and the evaluation results are compared with those of other related methods. The proposed method is shown to have the highest (100%) precision, recall and F1-score for anomaly detection, and have fairly high performance of anomaly classification. The contribution of this paper is to show that integrating the DAE, SMOTE, T-Link, and XGBoost schemes can achieve the highest or extremely high performance in the aspect of ICS anomaly detection and classification based on network traffic. The computational complexity and convergence analyses of the proposed method are also provided in this paper. Furthermore, the code implementing the proposed method is released for public access through IEEE Code Ocean so that the effectiveness and the applicability of the method can be validated.

INDEX TERMS Anomaly classification, anomaly detection, autoencoder, data imbalance, industrial control system, modbus, S7 communication, SMOTE, Tomek link, XGBoost.

I. INTRODUCTION

Due to the prevalence of the Industry 4.0 paradigm [1], more and more industrial machines and devices are interconnected through industrial field networks and even Internet. The industrial control system (ICS) [2], which integrates information technology (IT) and operational technology (OT), is used to monitor, control, and manage interconnected equipment in various industrial fields, such as power plants, water plants, oil refineries, public transportation systems, manufacturing factories, and so on [3]. The ICS brings many benefits such as high efficiency and good quality, but it also causes some potential shortcomings. For example, the ICS may suffer from cyberattacks [4], causing huge economic losses and even affecting the safety of personnel.

Many ICS cyberattacks occurred during last several years [5]. For example, Iran's nuclear power plant was attacked by Stuxnet, a malicious computer worm, in 2010. The worm propagated across the network and ruined almost 1000 nuclear centrifuges [6]. For another example, in 2018, a Taiwan chipmaker was halted by WannaCrypt malware, leading to the shutdown of many chip-fabrication factories connected through networks, resulting in a loss of approximately $256 million [7]. For yet another example, in 2021, an American oil pipeline system suffered a ransomware cyberattack affecting networked equipment managing the pipeline. All pipeline operations were stopped, causing oil supply chaos. Consequently, a ransom of $4.4 million was paid to restore the operations [8].

The rising number of ICS attacks has brought about the use of the intrusion detection system (IDS) in ICS networks [9]. An intrusion detection system is a hardware or software component continuously monitoring system behavior to report any anomalies, such as suspicious system events or network traffic abnormalities [10]. Anomalies are usually precursors to attacks or consequences of attacks. They should

The associate editor coordinating the review of this manuscript and approving it for publication was Nadeem Iqbal.

be detected and alarmed to administrators so that proper actions can be taken to secure the system. It is desirable to detect and even classify anomalies accurately and precisely for the administrators to make good decisions to enforce effective countermeasures against the anomalies.

This paper proposes an anomaly detection and classification method to detect and classify anomalies in an ICS by analyzing network traffic data of industrial field protocols like Modbus TCP (or Modbus, for short) [11] and S7 Communication (or S7 Comm, for short) [12]. The proposed method first uses the denoising autoencoder (DAE) [13] to reduce data noise and extract core features from data. It then employs the synthetic minority oversampling technique (SMOTE) [14] and the Tomek link (T-Link) [15] scheme to oversample and undersample data for addressing the data imbalance problem that the numbers of data in different classes differ significantly. Finally, extreme gradient boosting (XGBoost) [16] is used to build anomaly detection and classification models that can leverage the ensemble learning concept to avoid overfitting for achieving good performance.

A real-life railway industry dataset, called Electra [9], is used to evaluate the performance of the proposed method. The evaluation results are compared with those of other related methods [9], [17], whose performance is also evaluated by the Electra dataset. The existing methods proposed in [9] and [17] also apply resampling mechanisms to data to deal with the data imbalance problem. However, they have some limitations. Specifically, the methods proposed in [9] randomly select partial normal data so that the number of selected normal data equals the number of anomalous data. It is likely that some important features are missed due to the random selection of data, so the performance of the methods thus degrades. On the other hand, the method proposed in [17] uses the generative adversarial network model to generate synthetic anomalous data to mitigate the data imbalance problem. However, it requires sophisticated skills and much time to train the model for generating good synthetic data to achieve satisfactory anomaly detection performance.

The proposed method is shown to have the highest (100%) precision, recall and F1-score for the anomaly detection case. Moreover, it is also shown to have fairly high performance for the anomaly classification case. The contribution of this paper is to show that integrating the DAE, SMOTE, T-Link, and XGBoost schemes can achieve the highest or extremely high performance in the aspect of ICS anomaly detection and classification based on network traffic. Furthermore, the code implementing the proposed method is released for public access through IEEE Code Ocean so that the effectiveness and the applicability of the method can be validated.

In summary, the contribution of this paper is fourfold. First, it proposes a method integrating DAE, SMOTE, T-Link, and XGBoost mechanisms to detect or classify ICS anomalies by analyzing network traffic data. Specifically, DAE is for reducing data noise and extracting data features, SMOTE is for oversampling data, T-Link is for undersampling data, and XGBoost is for detecting/classifying anomalies. Second, the computational complexity analysis and convergence analysis of the proposed method are provided. Third, extensive experiments are conducted to evaluate the performance of the proposed method and the performance evaluation results are compared with those of other related methods. The comparison results show that the proposed method has the highest (100%) precision, recall and F1-score for anomaly detection, and has fairly high performance for anomaly classification. Fourth, the code implementing the proposed method is released through IEEE Code Ocean for validating the method's effectiveness and applicability.

The rest of this paper is organized as follows. Related research is introduced in Section II. The proposed method is elaborated in Section III, and its computational complexity and convergence analyses are provided in Section IV. The performance evaluation of the proposed method and its comparisons with related methods are shown in Section V. Finally, Section VI concludes this paper.

II. RELATED WORK

Gomez *et al.* investigated the ICS cybersecurity problem by proposing a methodology to generate a dataset called Electra [9]. The dataset is gathered from the network traffic of an electric traction substation used in the railway industry and can be publicly accessed [18]. An ICS testbed consisting of a supervisory control and data acquisition (SCADA) system, a firewall, a switch, a programmable logic controller (PLC) master, four PLC slaves, and several ICS devices is used for the data gathering. Devices of the ICS testbed communicate with each other with common industrial control protocols like Modbus [11] and S7Comm [12]. Since the protocols lack authentication, data encryption and data integrity checking, they are vulnerable to various attacks [19]. An extra computing device is deployed in the testbed with the man in the middle (MitM) setting to launch attacks for generating anomalous network packets. The extra device launches attacks and meanwhile records all packets, either normal or anomalous, for 12 hours to generate the dataset.

The Electra dataset contains 16M and 387M packet entries of Modbus and S7Comm traffic data, respectively. It is a very imbalanced dataset, as the percentages of the anomalous data are 5.2% for the Modbus traffic, and 1.42% for the S7Comm traffic. The anomalous data (or packets) are caused by three categories of attacks, namely reconnaissance attacks, false data injection attacks, and replay attacks. The attackers launch reconnaissance attacks to gain access to the industrial network for obtaining information of devices in the network and their associated services. The attackers usually plan next moves of attacks based on the information obtained. With false data injection attacks, attackers try to manipulate ICS devices. Attackers craft and transmit false data via control protocols to launch different types of attacks like command injection, response injection, parameter injection, and so on. Attackers launch replay attacks by retransmitting valid packets which they have ever intercepted in the network. Such attacks can mislead devices that receive the replayed packets.

The three categories of attacks concerning the Electra dataset are further classified into seven types, as shown in Table 1. Attackers launch the ''function code recognition attack'' by generating malicious packets to scan all possible function codes in the attacked PLC. Fake packets are created to perform the ''read attack'' or ''write attack'' on the effective memory address of the PLC. Packets are modified in the response of a slave device to launch the ''response modification attack'' or ''force error in response attack''. Packets are modified in command of a master device to launch the ''command modification attack''. As for the ''replay attack'', it is launched by retransmitting packets ever sent by the master or slave devices.

TABLE 1. The three categories and seven classes of attacks in the Electra dataset.

Attack Category	Attack Class	Packet Type
reconnaissance	function code recognition	packet creation
false data injection	response modification	packet modification
	force error in response	packet modification
	command modification	packet modification
	read data	packet creation
	write data	packet creation
replay	replay packet	packet creation

Gomez *et al.* also proposed supervised and unsupervised machine learning methods [9] to detect ICS anomalies by inspecting network packets of the Electra dataset. The supervised machine learning methods include the random forest (RF) model, the support vector machine (SVM) model, and the neural network (NN) model. The unsupervised machine learning methods include the one-class support vector machine (OCSVM) model and the isolation forest (IF) model. The hyperparameters and their possible values of different models are shown in Table 2. The grid-search mechanism is applied to select proper hyperparameter values to obtain the models having the best performance.

TABLE 2. The hyperparameters and their values of models investigated in [9].

Models	Hyperparameters	Values
RF	number of estimaters	100, 200, 300
SVM	C	0.01, 0.1, 1, 10, 100
	gamma	0.001, 0.01, 0.1, 1
NN	number of layers	1, 2, 3
	number of neurons per layer	128, 64, 32
OCSVM	nu	0.01, 0.1, 1
	gamma	0.01, 0.1, 1, auto
IF	number of estimators	100, 200, 300
	contamination	0.01, 0.005, 0.1, 0.5, auto

Among the above-mentioned models for Modbus traffic, the SVM model achieves the highest recall of 1 and the highest F1-score of 0.9876, whereas the RF model achieves the highest precision of 0.9877. For S7Comm traffic, the RF model achieves the highest recall of 1 and the highest F1-score of 0.9978, whereas the NN model achieves the highest precision of 0.9999.

Ning *et al.* [17] proposed a method based on the generative adversarial network (GAN) model [20] and the deep neural network (DNN) model for ICS anomaly detection. A GAN is a neural network with two adversarial parts that are trained together. One part is the generator trained to generate new (or fake) samples following the distribution of given real data samples. The other part is the discriminator trained to determine whether samples are real or not. The model training reaches equilibrium when the discriminator can no longer tell real samples from fake samples generated by the generator.

The GAN is used by Ning *et al.* [17] to generate additional anomalous data samples that are very similar to original anomalous samples from the perspective of statistical distribution to deal with the data imbalance problem. The newly generated samples are mixed with the original samples to be fed into a DNN model for detecting ICS anomalies. The Electra dataset for the Modbus protocol is employed to evaluate the performance of the proposed method. The authors used the GAN model to generate different numbers of additional anomalous samples to evaluate the recall metric of the DNN model. The GAN generator has 9 input units and 9 output units, whereas the GAN discriminator has 9 input units and 1 output unit. The generator (resp., discriminator) takes the linear (resp., sigmoid) function as the activation function, and takes the root-mean-square error (resp., logloss) function as the loss function. The DNN has 5 layers with different numbers of neurons with the rectified linear unit (ReLU) function [21] as the activation function. It is concluded that the GAN model can improve the recall of the DNN model from 0.89% to 0.94% by adding a small number (viz., 752213) of additional samples, and to 0.98% by adding a large number (viz., 1504463) of additional samples.

III. PROPOSED METHOD

This section describes the proposed method based on the DAE, SMOTE, T-Link, and XGBoost mechanisms. The input of the proposed method is the dataset of Modbus and S7Comm packet features and labels. The dataset is divided into the training and test datasets. The training dataset is used to train the DAE and the XGBoost models, whereas the test dataset, to verify the effectiveness of the trained models. The training data go through four major steps: data preprocessing, DAE training, SMOTE and T-Link resampling, and XGBoost training, as shown in Figure 1 (a). The trained DAE and the trained XGBoost are then applied to the test data without SMOTE and T-Link resampling to detect and classify anomalies, as shown in Figure 1 (b). Please refer to Appendix A for a running example of a test data item going through data preprocessing, DAE processing, and XGBoost processing. The major processing steps applied to the data are described in detail one by one in the following subsections.

A. DATA PREPROCESSING

Data preprocessing consists of the following routines. The first routine is redundancy removal to remove redundant data in the dataset. This is because in the industrial control system,

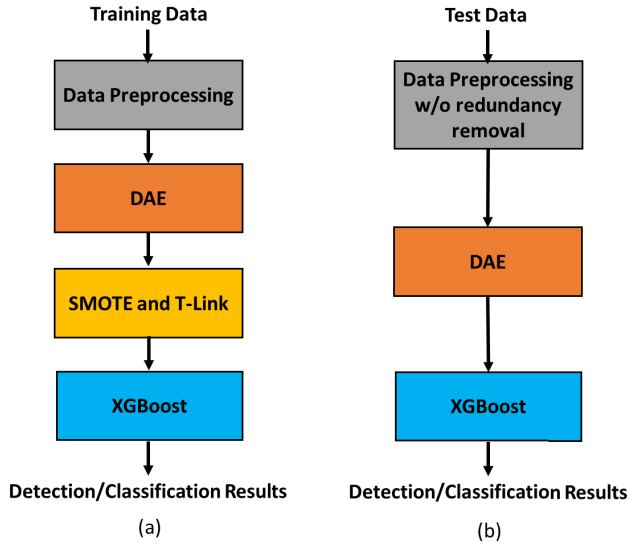


FIGURE 1. Major steps of the proposed method for (a) the training data and (b) the test data.

many control processes are re-executed over time. Consequently, there exist a large number of identical or redundant packets (or data) in the dataset. Redundant data should be removed for the sake of efficiency. This can be done by ignoring the ''time'' field of data to check if data are identical. For all identical data, only the first datum is retained, whereas others are regarded as redundancies to be removed. Note that the test data do not undergo the redundancy removal process, as every test datum should be classified separately.

The second routine is to apply one-hot encoding (OHE) to categorial data to increase one data dimension for each new category and to use 0 or 1 to indicate if a data item belongs to a certain category. OHE is useful for encoding categorial data without order relationship, such as MAC addresses and IP addresses. Moreover, the second routine is also to employ label encoding to encode categorial data having no or little order relationship, such as function codes, memory addresses, and data to be written or read. Unlike OHE, label encoding transforms categorial data into integers ranging from 0 to the number of categories minus one without increasing data dimensionality [22].

The third routine is to standardize scalar data so that they follow the standard normal distribution with the mean of 0 and the standard deviation of 1. This is achieved by calculating the mean and the standard deviation of the original data and afterward subtracting each original data value by the mean and then dividing the subtracted value by the standard deviation.

B. DAE TRAINING

After data are preprocessed, they are used to train the DAE model [13] to reduce noise in data and to extract core features of the data. A DAE is a special type of the autoencoder (AE) model [23] consisting of layers of connected neurons. A neuron takes a vector x as input and produces output y according to the following Equation (1).

$$y = \sigma(w \cdot x^T + b), \tag{1}$$

where w is the vector of neuron connection weights, x^T is the transpose of input vector x, b is a bias value, and σ is the activation function (e.g., the sigmoid function). Figure 2 shows the structure of the DAE model, as elaborated below.

As shown in Figure 2, the right portion of an DAE is actual an AE consisting of the encoder, the code, and the decoder. The DAE first adds artificial noises into the original input x to have the noisy input $\tilde{x}$. The noisy input is then fed into the encoder to generate the code, which in turn is transformed by the decoder to be the reconstructed input x'. The reconstructed input x' is intended to be as close to the original input x as possible. This can be achieved by tuning the neuron connection weights to minimize the mean square error (MSE) of a batch of inputs and reconstructed inputs with the error backpropagation mechanism, the gradient descent algorithm, and an optimizer like the adaptive moment estimation (Adam) [24]. Consequently, the DAE can be used to reduce noise of input data and extract the data feature as a code. The code usually has much smaller dimensionality than the original data. The trained DAE can then be applied to each datum to represent it as a code to reduce noise and dimensionality for the purpose of feature extraction. However, the method proposed in this paper uses the output of the trained DAE as the extracted features of the original datum. That is to say, DAE is used for reducing noise and extracting features but not for reducing dimensionality in the proposed method.

C. SMOTE AND TOMEK-LINK RESAMPLING

Resampling is necessary for training models with an imbalanced dataset, where the majority class samples (or data) significantly outnumber the minority class samples for the binary-class classification case. Therefore, after all input data are represented as codes, the codes are oversampled by the synthetic minority oversampling technique (SMOTE) [14] and undersampled by the Tomek-link (T-Link) mechanism [15].

SMOTE is an approach to improve the traditional random oversampling mechanism that repeatedly copies arbitrary samples (or data) from the minority class. SMOTE generates new synthesized samples based on the interpolation of two minority class samples. Note that below a ''minority sample'' stands for a ''minority class sample'', whereas a ''majority sample'' stands for a ''majority class sample''.

Let the number of minority samples be m and the oversampling rate be n, $n = 1, 2, \ldots$, which is the number of times of oversampling each minority sample. For a minority sample x_i, $1 \leq i \leq m$, the steps for SMOTE to synthesize n new samples based on x_i are as follow. The reader can refer to Figure 3 for the illustration of SMOTE oversampling.

Step 1: For a minority sample x_i, $1 \leq i \leq m$, find the set S_i of k minority samples that are closest to x_i by applying the k-nearest neighbor (k-NN) algorithm to x_i.

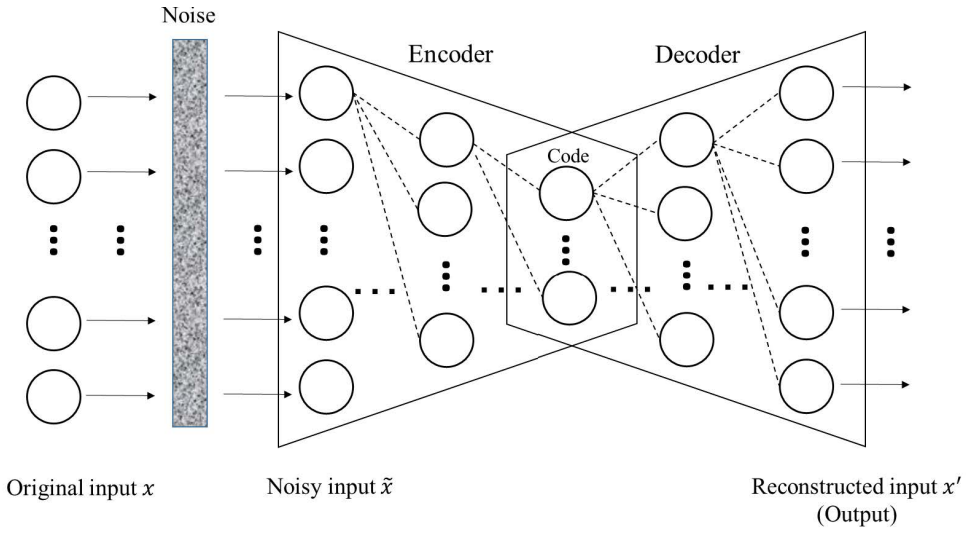


FIGURE 2. The structure of a denoising autoencoder (DAE).

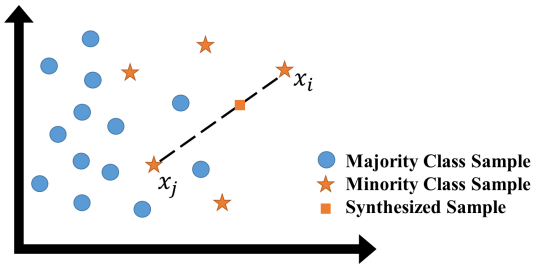


FIGURE 3. Illustration of SMOTE oversampling minority class samples.

Step 2: Randomly select a minority sample x_j from S_i and generate a new synthesized sample x_{new} according to the following equation:

$$x_{new} = x_i + \delta \times (x_j - x_i),$$

where δ is an arbitrarily selected value between 0 and 1.

Step 3: If the number of new samples synthesized on the basis of x_i is less than n, then go to Step 2.

T-Link is a mechanism to improve traditional undersampling technique which randomly removes a certain portion of majority samples. T-Link focuses on the T-link pair of a minority sample and a majority sample. The majority sample of every T-link pair is then removed.

For a minority sample x_i, $1 \leq i \leq m$, the steps to remove majority class samples based on x are as follow:

Step 1: Find the majority sample x_j that is closest to the minority sample x_i.

Step 2: Calculate $d(x_i, x_j)$, which is the Euclidean distance between x_i and x_j.

Step 3: If there exists no sample x_k such that $d(x_i, x_k) < d(x_i, x_j)$ or $d(x_k, x_j) < d(x_i, x_j)$, then (x_i, x_j) is called a T-link pair.

Step 4: If (x_i, x_j) is a T-link pair, then remove x_j from the majority class.

The above-mentioned steps can be repeated to remove majority samples. The readers are referred to Figure 4 for the illustration of T-Link that undersamples data in the majority class. Unlike traditional undersampling techniques which randomly remove majority samples, T-Link removes only majority samples of T-link pairs. Consequently, T-Link can effectively make the boundary between the majority class and the minority class more distinguishable and avoid the problem that important information is lost due to the random removal of majority samples.

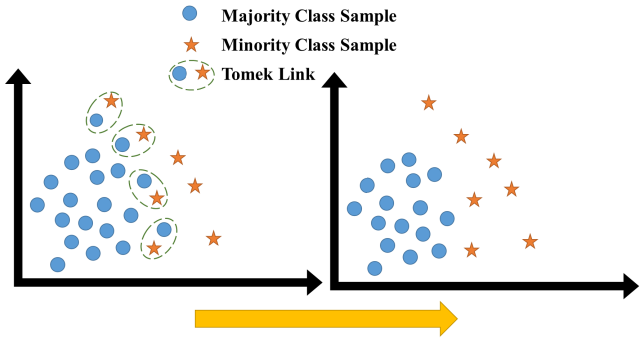


FIGURE 4. Illustration of T-Link undersampling majority class samples.

Using SMOTE to generate minority samples may lead to the problem that the generated minority samples may be surrounded by some majority samples. It is thus hard to distinguish the minority samples from the majority samples. Using T-Link right after using SMOTE can mitigate the problem. Therefore, the method proposed in this paper uses the combination of SMOTE and T-Link to sequentially oversample and undersample data.

D. XGBoost TRAINING

After all input data are represented as DAE codes to be resampled by SMOTE and T-Link, the resampled codes are used to train the XGBoost model for classifying the input data. Below, we introduce the XGBoost model and describe how to train the model with resampled DAE codes.

XGBoost [16] is an ensemble learning model that improves the gradient boosting decision tree (GBDT) mechanism [25]. Its basic concept is to combine many weak estimators to derive a strong estimator. Specifically, it uses an additive strategy to use multiple decision trees (DTs) or classification and regression trees (CARTs) [26] to obtain a model that can predict (or classify) the target labels by selecting proper attributes (or features) to split at appropriate values for the purpose of classification or regression. Specifically, a DT is a tree structure in which each internal node represents a ''check'' of particular conditions on an attribute, each branch represents the check outcome, and each leaf node represents a class label. Moreover, XGBoost also employs the L_2-norm regularization [27] to mitigate the overfitting problem.

XGBoost first builds a DT for predicting the target label by optimizing the objective function. Then, the second DT is built to predict the residual error (i.e., the second target label) between the first DT's prediction and the target label. Generally, a t^{th} DT is built to predict the residual error between the $(t-1)^{th}$ DT's prediction and the $(t-1)^{th}$ target label.

For a given dataset $\mathcal{D} = \{(x_i, y_i)\}$ with n samples and m attributes, where $|\mathcal{D}| = n$, $x_i \in \mathbb{R}^m$ is a data sample, $y_i \in \mathbb{R}$ is a label, and $1 \le i \le n$. Let $f_t(x_i)$ be the function associated with the t^{th} DT, and $\hat{y}_i^{(t)}$ be the prediction of the i^{th} sample of the t^{th} DT. We have the two following Equations (2) and (3).

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i) = \sum_{k=1}^{t} f_k(x_i), f_k \in \mathcal{F} \tag{2}$$

$$Obj^{(t)} = \sum_{i=1}^{n} loss(y_i, \hat{y}_i^{(t)}) + \sum_{k=1}^{t} \Omega(f_k) \tag{3}$$

In Equation (2), $\mathcal{F} = \{f(x) = w_{q(x)}\}$ in the space of DTs, where $q : \mathbb{R}^m \to T$, and $w \in \mathbb{R}^L$. Specifically, q represents the function associated with the DT mapping a sample of m attributes to a leaf index of $1, \ldots,$ or T, and w represents the vector of leaf weights. In Equation (3), $Obj^{(t)}$ is the objective function to be optimized (or minimized), $loss(y_i, \hat{y}_i^{(t)})$ is the loss function that measures the difference between the target label y_i and the prediction $\hat{y}_i^{(t)}$, and $\Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|_2^2$ is the function to penalize the complexity of function f associated with the DT, where γ and λ are two constants, and $\|w\|_2^2$ is the square of L_2-norm of w, the vector of leaf weights.

IV. COMPUTATIONAL COMPLEXITY AND CONVERGENCE ANALYSES

In this section, the computational complexity analysis and convergence analysis of the proposed method are demonstrated. The proposed method employs the DAE model to reduce data noise and extract data core features. The DAE model is a special type of neural networks. As stated in [28], the computational complexity to train a neural network model with w weights is linear in w. In summary, the DAE model has the computational complexity of $O(w)$, where w is the number of DAE weights, including biases. Specifically, for a DAE with a fully connected neural network structure of l layers with v_i neurons in layer i, $1 \le i \le l$, the number w of weights is proportional to $\sum_1^{l-1} (v_i v_{i+1} + v_{i+1})$.

The proposed method utilizes the SMOTE and the T-Link mechanisms to oversample and undersample data. As shown in [29], the computational complexity of the SMOTE mechanism is $O(n \log n)$. Moreover, the proposed method uses T-Link right after using SMOTE to find and remove a majority sample for each minority sample with the concept of the T-link pair. The total computational complexity of using T-Link after using SMOTE is also $O(n \log n)$, as shown in [30].

The proposed method uses XGBoost to classify data samples. As shown in [16], training an XGBoost model and using a trained XGBoost model for data classification respectively takes $O(tdx \log n)$ and $O(td)$ computational complexity, where t is the number of the decision trees, d is the maximum depth (or height) of the trees, x is the number of non-missing entries in the training data, and n is the number of training data samples.

The convergence analysis of the DAE model is shown below. Since the convergence of training neural network models is closely related to the adopted optimizer, we analyze the convergence of the Adam optimizer that is adopted in this paper to train the DAE of our proposed method.

As mentioned earlier, Adam stands for adaptive moment estimation [24]. It is widely adopted as the optimizer for training neural networks due to its good performance. Adam estimates the first-order and the second-order gradient moments to adaptively update individual learning rates for different neural network weights [31]. Below are the calculations used by Adam to update every weight w_t of a neural network at iteration t for $0 \le t \le T$.

$$g_t = \nabla_w f_t(w_{t-1}) \tag{4}$$

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{5}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{6}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{7}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{8}$$

$$w_t = w_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{9}$$

In the above equations, g_t is the gradient of the objective function (or loss function) f_t, m_t is the biased first moment (or mean) estimate, v_t is the biased second raw moment (or uncentered variance) estimate, β_1 and β_2 are exponential decay rates with values in [0,1), $\hat{m}_t$ is the bias-corrected first moment estimate, $\hat{v}_t$ is the bias-corrected second raw moment estimate, α is the learning rate (or step size), and ϵ is a

sufficiently small number. As suggested in [24], good default parameter settings are as follows: $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, $m_0 = 0$, and $v_0 = 0$.

As shown in [24], equations (7), (8), and (9) show above can be replaced with the following two equations:

$$\alpha_t = \alpha\sqrt{1-\beta_2^t}\,/(1-\beta_1^t) \tag{10}$$

$$w_t = w_{t-1} - \alpha_t\, m_t\, /\, (\sqrt{v_t} + \epsilon) \tag{11}$$

Although the paper [32] shows a counterexample of a simple convex optimization setting for which Adam does not converge, some papers [33]–[35] still provide the sufficient conditions of convergence and the convergence rate for Adam with special settings. The paper [33] shows the sufficient conditions of convergence for generic Adam that iteratively updates β_1 (shown as β_t below) and β_2 (shown as θ_t below). To be more specific, generic Adam is convergent if parameter sequences $\{\alpha_t\}$, $\{\beta_t\}$, and $\{\theta_t\}$ satisfy all the following conditions [33]:

1. $\beta_t \le \beta_0 < 1$;
2. $0 < \theta_t < 1$, and θ_t is non-decreasing;
3. $\alpha_t\, /\sqrt{1-\theta_t}$ is "almost" non-increasing;
4. $(\sum_{t=1}^{T} \alpha_t\sqrt{1-\theta_t})\, /\, (T\alpha_T) = o(1)$.

We adapt the above conditions for original Adam to keep β_t as a constant of β_1, and to keep θ_t as a constant of β_2. Therefore, the sufficient conditions for original Adam, based on equations (4), (5), (6), (10), and (11), to converge are shown as follows:

1. $\beta_1 < 1$;
2. $0 < \beta_2 < 1$;
3. $\alpha_t\, /\sqrt{1-\beta_2}$ is "almost" non-increasing;
4. $(\sum_{t=1}^{T} \alpha_t\sqrt{1-\beta_2})\, /\, (T\alpha_T) = o(1)$.

The paper [34] shows Adam has the convergence rate $O(1/\sqrt{T})$ in the non-convex stochastic optimization setting with the batch size of the same order of T. That is to say, Adam will converge to a stationary point with an error in the order of $O(1/\sqrt{T})$ after T iterations. Moreover, the paper [35] shows that Adam with a learning rate $\alpha = 1/\sqrt{T}$ and a momentum parameter of squared gradients $\beta_2 = 1-(1/T)$ has the convergence rate $O(\ln T/\sqrt{T})$.

V. PERFORMANCE EVALUATION AND COMPARISON

A. THE DATASET

The Electra dataset [18] reported in [9] for ICS cybersecurity research is used to evaluate the performance of the proposed method. The Electra dataset has two sub-datasets: the Electra Modbus dataset for the Modbus protocol and the Electra S7Comm dataset for the S7Comm protocol. Each Electra data entry contains packet-level features like the function code and the MAC/IP addresses of the two communicating parties of the Modbus or the S7Comm packets. Table 3 shows the features, descriptions, and the data types of the Electra dataset entry.

Each data entry in the Electra dataset contains only a single operation. Entries are labeled as normal or anomalous with

TABLE 3. Data features, descriptions, and types of the Electra dataset.

Features	Descriptions	Types
time	timestamp	string
smac	source MAC address	string
dmac	destination MAC address	string
sip	source IP address	string
dip	destination IP address	string
request	indicating whether the packet is a request from the master to the slave	boolean
fc	function code	integer
error	indicating whether errors exist in reading/writing operations	boolean
madd	memory address for performing read/write operation	integer
data	data for performing read/write operation	integer
label	indicating either normality or the class of the attack	string

TABLE 4. Class percentages and sizes in the Electra Modbus dataset and the Electra S7Comm dataset.

Electra Modbus Dataset		
Class	**Percentage**	**Size**
Normal	94.8%	15444940
Response modification attack	0.1%	16353
Force error in response attack	0.007%	1129
Read attack	4.83%	785961
Write attack	0.06%	9417
Replay attack	0.006%	897
Function code recognition attack	0.19%	30580
All classes	100%	16289277
Electra S7Comm Dataset		
Class	**Percentage**	**Size**
Normal	98.58%	38161328
Response modification attack	0.04%	15484
Force error in response attack	0.42%	162581
Read attack	0.23%	89033
Write attack	0.57%	220646
Replay attack	0.007%	2710
Command modification attack	0.15%	58065
All classes	100%	38709847

different classes of attacks. The percentage and size of every possible class is demonstrated in Table 4. For the Modbus protocol, the possible classes of labels are normal, the response modification attack, the force error in response attack, read attack, write attack, replay attack, and the function code recognition attack. For the S7Comm protocol, the possible classes of labels are normal, the response modification attack, the force error in response attack, read attack, write attack, replay attack, and the command modification attack. Note that the data entries for the Modbus and the S7Comm protocols have six common classes and one different class. It can be observed from Table 4 that the Electra dataset is a very imbalanced dataset. The percentage of the normal class is 94.8% for the Electra Modbus dataset; 98.58%, for the Electra S7Comm dataset.

B. EVALUATION METRICS

When dealing with highly imbalanced data for performing classification, it is trivial to get a high accuracy by always classifying data into the majority class [36]. Since the Electra

dataset used for performance evaluation is extremely imbalanced, the accuracy metric, as defined in Equation (12), is not adopted for the evaluation. Instead, the precision, the recall, and the F1-score, which are defined in Equations (13), (14), and (15), are used as metrics for performance evaluation.

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \tag{12}$$

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{13}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{14}$$

$$\text{F1-score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{15}$$

In Equations (12), (13), and (14), TP, FP, TN, and FN stand for the numbers of true positive, false positive, true negative, and false negative detections. In the context of this paper, a positive detection is to classify a packet to be abnormal or anomalous, whereas a negative detection is to classify a packet to be normal. TP (resp., FP) is the number of positive detections for actually anomalous (resp., normal) packets. TN (resp., FN) is the number of negative detections for actually normal (resp., anomalous) packets.

It is desirable to have as high as possible precision, recall and F1-score, whose values are all between 0 and 1. When there is no false positive detection (i.e., FP is 0), the highest precision of 1 is achieved. When there is no false negative detection (i.e., FN is 0), the highest recall of 1 is achieved. However, it is hard to achieve high precision and high recall at the same time. There are usually trade-offs between precision and recall. The F1-score, which considers precision and recall at the same time, can thus be used as a good metric for performance evaluation. When precision and recall are both 1, the highest F1-score of 1 is achieved.

The above-mentioned descriptions are for the case of binary-class classification metrics. For the case of multi-class classification, this paper adopts the macro averaging scheme to derive metrics. In general, for a k-class classification case to classify data into class 1, class 2, ..., and class k, the macro-averaged metric is calculated as follows. Taking class 1 as the positive class and others as the negative class to calculate the metric for class 1, ..., taking class k as the positive class and others as the negative class to calculate the metric for class k, and finally averaging metrics of all classes to derive the macro-averaged metric.

C. PERFORMANCE EVALUATION

The Electra dataset is first divided into a training dataset of 80% training data, and a test dataset of 20% test data for evaluating the performance of the proposed method. The performance evaluation is for the Modbus protocol and the S7Comm protocol, and has the following two cases: the anomaly detection (or binary-class classification) case and the anomaly multi-class classification case.

The training data first undergo redundancy removal. This is because ICS control processes frequently repeat identical actions, causing a great number of identical or redundant packets. Note that the paper [9] performs redundant packet removal for all data. However, this paper performs redundant packet removal only for the training data, but not for the test data. In practical applications, not all of data are available in advance. Hence, test data do not undergo redundancy removal.

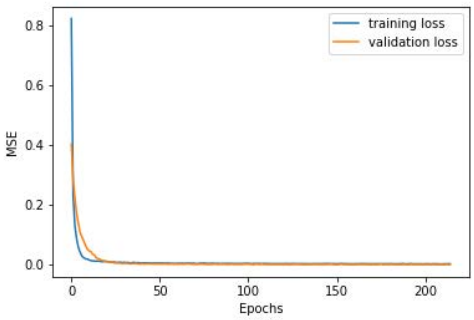
The training data are used to train the DAE model. The hyperband algorithm [37] is used to tune hyperparameters of the DAE to derive the best model. Major DAE hyperparameters and their possible values are shown in Table 5. Note that the asterisked and underlined values in Table 5 represent the values selected by the hyperband algorithm for Modbus traffic and S7Comm traffic, respectively. The selective values are tuned for the multi-class classification case; they are also applied for the binary-class classification case, though.

TABLE 5. DAE hyperparameters and values used by the proposed method.

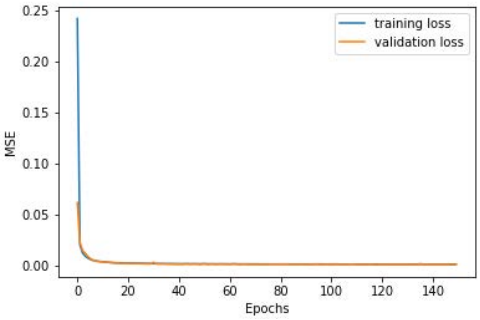
Hyperparameters	Values
Number of layers in the encoder/decoder	1*, 2, 3, 4, 5, 6
Number of neurons in the first layer of the encoder	64*, 128, 256, 512
Activation function	SELU*, ReLU
Initializer	Glorot Uniform, Glorot Normal*, He Uniform, He Normal, LeCun Uniform, LeCun Normal
Optimizer	Adam*, SGD, RMSProp

The DAE model for Modbus traffic has the following neural network structure: the input layer of 21 neurons with batch normalization, three hidden layers of 64, 32, and 64 neurons, all with batch normalization, and the output layer of 21 neurons. Note that each encoder (resp., decoder) layer of the DAE is assumed to have half (or twice) neurons as its previous layer. The activation function of each neuron is the scaled exponential linear unit (SELU) function [38]. The initializer is Glorot normal [39], and the optimizer is Adam [24]. The DAE model for the S7Comm protocol case has similar structure except that the input layer and the output layer have 24 neurons. Note that batch normalization [40] is a mechanism to improve neural network training through input re-normalization for each layer of neurons.

The input data are fed into the DAE to extract features as DAE codes. The SMOTE and the T-Link mechanisms are then applied to the codes (or samples) for oversampling and undersampling them. Note that the proposed method does not apply the two resampling mechanisms for the anomaly detection case, as its performance is good enough without them. However, the proposed method indeed applies the two mechanisms for the multi-class classification case. The SMOTE mechanism adopts $k = 5$ for the k-NN algorithm to oversample minority samples. For specific minority classes, a target number of minority samples is set, and the undersampling continues until the number of the minority samples approximates the target number. For example, the target numbers of the "replay attack" and "read attack" samples are



(a) DAE training for Modbus traffic



(b) DAE training for S7Comm traffic

FIGURE 5. The loss-epoch curves of DAE training for the Electra dataset.

both set to 5000 for Modbus traffic. For another example, the target number of the ''replay attack'' samples is set to 700 for S7Comm traffic. After the SMOTE oversampling, the T-Link mechanism is then utilized to find a T-link pair for every minority sample to remove the majority sample from the pair to undersample majority samples. Afterwards, the resampled data are then used to train the XGBoost model for classifying input data.

Fixed hyperparameter settings are used for training the XGBoost model. Some of important settings are as follows. The number of estimators (DTs) is 1000, the maximum depth of tree is 6, and the learning rate is 0.300000012. The parameter settings are applied for both the binary-class classification and the multi-class classification cases, and for both Modbus traffic and S7Comm traffic.

After the training data are used to train the DAE and the XGBoost models, the test data are then applied to the trained DAE and the trained XGBoost models for classifying the data packets. Note that the test data do not undergo the redundancy removing and resampling, as in practice not all test data are known in advance. However, the test data do undergo the standardization process by using the mean and the standard deviation of the training data.

Figure 5 shows the loss-epoch curves in training the DAE model for the Electra dataset of Modbus traffic and S7Comm traffic. The curves show that MSE losses converge very soon in the training history. The early-stopping mechanism is further employed to save training time and to avoid the overfitting problem. The DAE training stops at the 323^{rd} (resp., 111^{st}) epoch for the Modbus (resp., S7Comm) traffic data. Moreover, Figure 6 shows the loss-epoch curves in training the XGBoost model for the Electra dataset of Modbus traffic and S7Comm traffic. The curves show that the cross-entropy losses converge soon in the training history. The early-stopping mechanism is also employed to save training time and to avoid the overfitting problem. The XGBoost training stops at the 64^{th} (resp., 88^{th}) epoch for the Modbus (resp., S7Comm) traffic data in the binary-class classification case. It stops at the 52^{nd} (resp., 73^{rd}) epoch for the Modbus (resp., S7Comm) traffic data in the multi-class classification case.

Table 6 shows the time overheads for training the DAE and XGBoost models and for performing SMOTE and T-Link. It also shows the time overheads for analyzing a test datum for anomaly detection/classification. Note that a test datum goes through data preprocessing, DAE feature extraction, and XGBoost detection/classification. As shown in Table 6, it takes only few milliseconds to analyze a test datum for anomaly detection/classification, which makes the proposed method meet the real time requirement.

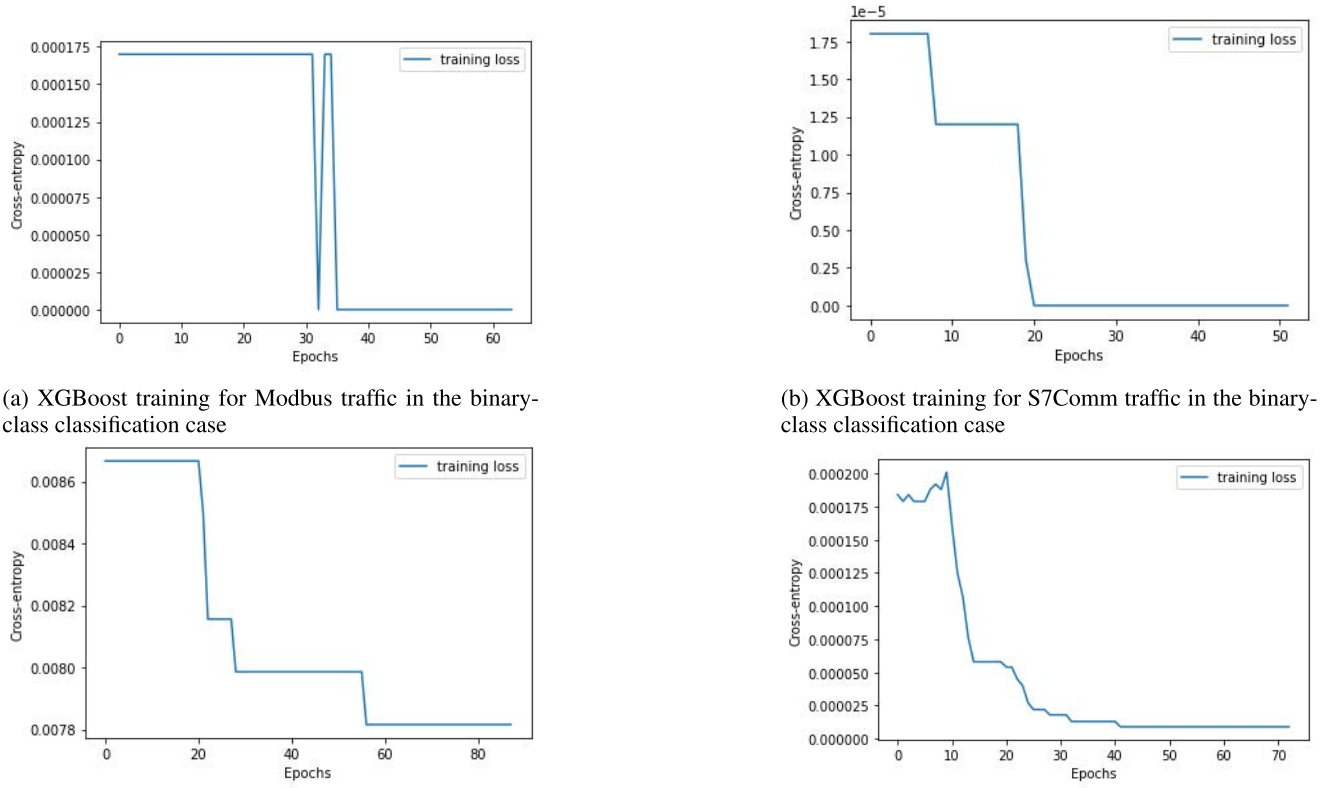
The device used to train the DAE and XGBoost models and to analyze a test datum has the following specifications.

- CPU : Intel® Xeon® Dual-core CPU E5-2630 v4 @ 2.20 GHz and 2.20 GHz
- RAM : 256 GB
- GPU : Two NVIDIA GeForce GTX 1080

D. PERFORMANCE COMPARISONS

The performance evaluation results of the proposed method for the binary-class classification case (i.e., anomaly detection case) are shown in Figure 7 in the form of confusion matrices. The results of the proposed method are quite good for both the Modbus and the S7Comm datasets. The precision, recall and F1-score are all 100%. That is to say, the proposed method is a perfect classifier whose classifications are all correct. Table 7 shows the anomaly detection performance comparisons of the proposed method and other related ones like the SVM, OCSVM, RF, IF, NN, and GAN+DNN methods investigated in [9], [17] in terms of the precision, recall and F1-score. Note that the GAN+DNN method does not use the S7Comm dataset to evaluate its performance. Therefore, it is not included in the comparisons for the S7Comm dataset in Table 7.

The performance evaluation results of the multi-class classification case are shown in Figure 8 in the form of confusion matrices. Table 8 shows the multi-class classification performance of the proposed method in terms of the precision, recall and F1-score. To the best of our knowledge, there is no other research that proposes anomaly multi-class



(a) XGBoost training for Modbus traffic in the binary-class classification case

(b) XGBoost training for S7Comm traffic in the binary-class classification case

(c) XGBoost training for Modbus traffic in the multi-class classification case

(d) XGBoost training for S7Comm traffic in the multi-class classification case

FIGURE 6. The loss-epoch curves of XGBoost training for the Electra dataset.

TABLE 6. The time overheads (in seconds) for training DAE, for performing SMOTE and T-Link, for training XGBoost with training data, and for analyzing a test datum for anomaly detection and classification.

Electra Datasets	Training Data				Test Datum	
	for DAE	for SMOTE+T-LINK	for XGBoost (detection)	for XGBoost (classification)	for detection	for classification
Modbus	26.282858848	0.389975786	2.052809957	84.843178987	0.001902580	0.002983331
S7Comm	142.599220275	169.337571620	175.019133090	1131.000727415	0.003566265	0.004170179

TABLE 7. Anomaly detection performance comparisons of the proposed methods and related methods.

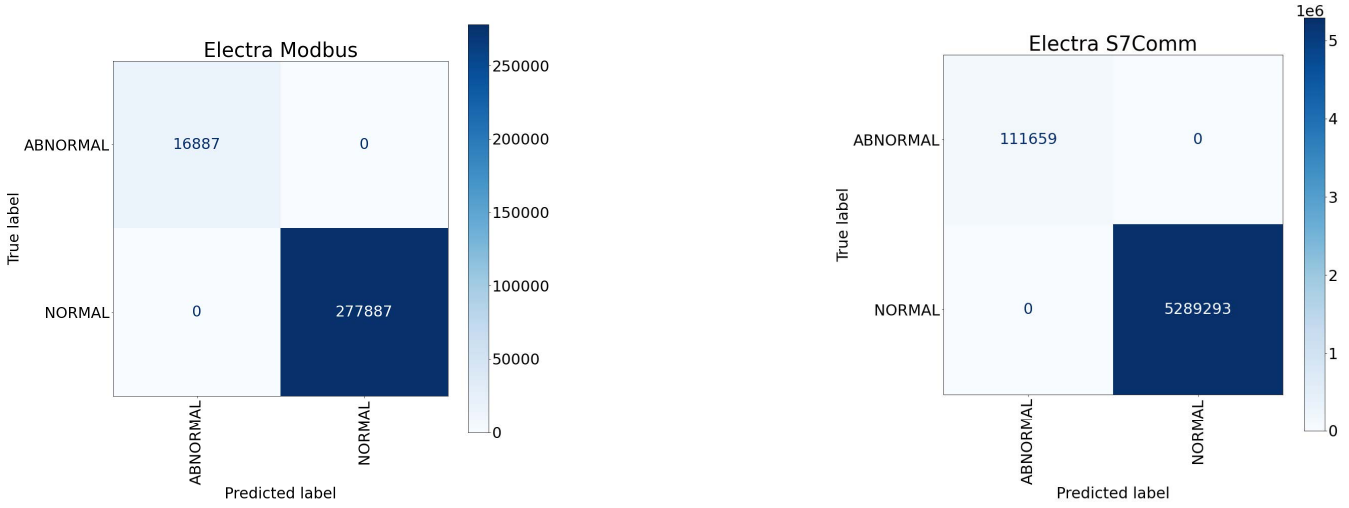
Dataset	Model	Precision	Recall	F1-Score
Electra Modbus	SVM [9]	0.9756	1	0.9876
	OCSVM [9]	0.9862	0.9856	0.9859
	RF [9]	0.9877	0.9871	0.9874
	IF [9]	0.8739	1	0.9327
	NN [9]	0.9692	1	0.9843
	GAN+DNN [17]	-	0.98	-
	Proposed Method	1	1	1
Electra S7Comm	SVM [9]	0.9949	1	0.9974
	OCSVM [9]	0.9961	1	0.9971
	RF [9]	0.9956	1	0.9978
	IF [9]	0.9886	1	0.993
	NN [9]	0.9999	0.9919	0.9959
	Proposed Method	1	1	1

TABLE 8. Anomaly multi-class classification performance of the proposed method.

Method	Dataset	Precision	Recall	F1-score
Proposed Method	Electra Modbus	0.9999	0.9767	0.9850
	Electra S7Comm	0.9999	0.9999	0.9999

classification methods of which performance is evaluated on the basis of the Electra dataset. Thus, Table 8 only shows the performance of the proposed method. There is still room to improve the performance of the proposed method for the multi-class classification case. Specifically, some anomalies caused by read attacks are misclassified as those caused by replay attacks in Modbus traffic. On the other hand, some anomalies caused by replay attacks are misclassified as those caused by write attacks in S7Comm traffic. This is due to the fact that anomalies caused by read attacks and replay attacks have very similar patterns in network traffic, a fact also reported in [9].

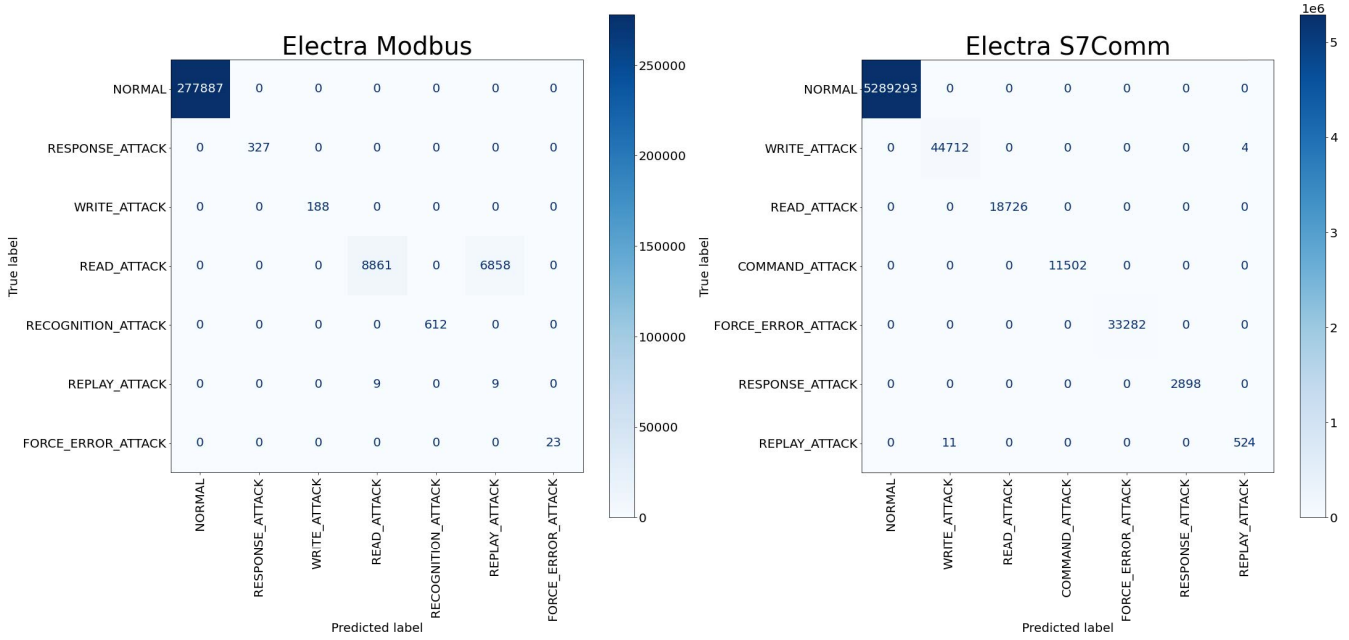
As just shown, the proposed method integrating DAE, SMOTE, T-Link, and XGBoost has the best performance or very good performance results that are superior to



(a) Modbus binary-class classification

(b) S7Comm binary-class classification

FIGURE 7. **Confusion matrices of the proposed method for the binary-class classification case.**



(a) Modbus multi-class classification

(b) S7Comm multi-class classification

FIGURE 8. **Confusion matrices of the proposed method for the multi-class classification case.**

those of related methods. Below we show some performance results generated during the early stage of our investigation to reveal the reason for choosing such integration. Table 9 shows performance results of various combinations of different schemes, including XGBoost alone, AE+XGBoost, DAE+XGBoost, DAE+SMOTE+T-Link+XGBoost, DAE+SMOTE+T-Link+RF, and AE+SMOTE+T-Link+XGBoost, based on the Electra Modbus and S7Comm datasets for the anomaly classification. Note that like the methods proposed in [9], the integrated methods shown in Table 9 use the test data undergoing the process of redundancy removal. This is because the methods developed in our early investigation stage follow the practice of the existing methods proposed in [9]. Therefore, the results in Table 9 are a little different from those shown in Table 8.

According to our previous research [41], we know that AE and RF are insensitive to imbalanced data and thus can deal with them properly. DAE and XGBoost are advanced schemes related to AE and RF, respectively. By performance results shown in Table 9, we further know that DAE and XGBoost can even achieve a little better performance than AE and RF. In practice, XGBoost alone can achieve very

TABLE 9. Performance results for different methods integrating various schemes.

Electra Modbus Dataset			
Integration of schemes	**Precision**	**Recall**	**F1-score**
XGBoost	0.9968	0.9968	0.9968
AE+XGBoost	0.9970	0.9970	0.9970
DAE+XGBoost	0.9966	0.9970	0.9968
DAE+SMOTE+T-Link+XGBoos	0.9987	0.9979	0.9982
DAE+SMOTE+T-Link+RF	0.9980	0.9974	0.9977
AE+SMOTE+T-Link+XGBoost	0.9985	0.9978	0.9980
Electra S7Comm Dataset			
Integration of schemes	**Precision**	**Recall**	**F1-score**
XGBoost	0.9998	0.9998	0.9998
AE + XGBoost	0.9997	0.9998	0.9997
DAE+XGBoost	0.9998	0.9998	0.9998
DAE+SMOTE+T-Link+XGBoost	0.9998	0.9998	0.9998
DAE+SMOTE+T-Link+RF	0.9997	0.9996	0.9997
AE+SMOTE+T-Link+XGBoost	0.9998	0.9998	0.9998

good performance. In addition to XGBoost, using DAE can improve the performance a little. Furthermore, additionally using SMOTE and T-Link can still improve the performance, although the improvement is not very significant. Therefore, the proposed method adopts the integration of DAE, SMOTE, T-Link, and XGBoost for detecting and classifying anomalies.

VI. CONCLUSION AND FUTURE WORK

An ICS anomaly detection and classification method is proposed in this paper to detect and classify anomalies based on network traffic data of industrial field protocols like Modbus and S7Comm. The proposed method uses the DAE, SMOTE, T-Link, and XGBoost mechanisms to achieve good detection and classification performance. The Electra dataset is used to evaluate the performance of the proposed method, and the performance evaluation results are compared with those of related methods, such as SVM, OCSVM, RF, IF, NN, and GAN+DNN. The proposed method achieves the highest (100%) precision, recall and F1-score for the case of binary-class classification. It also achieves very high performance for the case of multi-class classification.

Besides binary-class and multi-class classification performance, this paper evaluates the convergence and time overhead performance for the proposed method. The loss-epoch curves of DAE and XGBoost training history are provided to show the convergence of losses. The time overheads consumed by the DAE, SMOTE, T-Link, and XGBoost mechanisms are also provided. However, this paper has a limitation that it does not provide convergence analysis and time complexity analysis for the proposed method. This is because such analyses involve many factors, like tunable hyperparameters and variant distributions of input data. Some factors are even dynamically changing (e.g., adaptive learning rates), unknown in advance or hard to model (e.g., input data distributions).

The code implementing the proposed method is released for public access through IEEE Code Ocean to facilitate readers to apply the method to real world cases. The effectiveness of the proposed method has been validated by the Electra dataset that is based on the Modbus and the S7Comm protocols. The two protocols are used or supported by many prevalent ICS devices and systems. For example, most SCADA systems support the Modbus protocol, and numerous PLC devices support the S7Comm protocol. The proposed method is thus suitable for detecting and/or classifying anomalies in practical ICSs adopting such SCADA systems and PLC devices.

In addition to the Electra dataset, many datasets related to ICS security are available publicly. Typical ICS security datasets include the secure water treatment (SWaT) dataset [42], water distribution (WADI) dataset [43], electric power and intelligent control (EPIC) dataset [44], gas pipeline dataset [45], power system dataset [46], water storage dataset [47], and power grid dataset [48], etc. The datasets are associated with different scenarios, applications, communication protocols, data features, and classes of attacks (or anomalies). For example, the protocols related to the datasets are the Common Industrial Protocol (CIP), EtherNet/IP, IEC 61850, Remote Terminal Unit (RTU) serial communications, and Distributed Network Protocol 3 (DNP3). Possible attacks considered are the altering sensor reading attack, relay trip command injection attack, disabling relay function attack, denial of service (DoS) attack, ARP spoofing attack, evil twin attack, reconnaissance attack, command injection attacks (e.g., the address scan, function scan, and illegal setpoint attacks in the gas pipeline dataset, as well as the state command injection, parameter command injection, and function command injection attacks in the water storage dataset), and data/response injection attacks (e.g., the negative value, burst values, fast change, single data injection, slow change, value wave injection, and setpoint value injection attacks in the gas pipeline dataset, along with the negative level, above high setpoint, below low setpoint, random, and replay attacks in the water storage dataset, as well as the naïve malicious response injection, and complex malicious response injection in the water storage dataset). Possible data features are the phase current magnitude measured at each relay, relay status for each relay, snort alert status for each relay, and control panel remote trip status in the power system dataset, and command address, response address, command memory, response memory, command memory count, control mode, control scheme, subfunction code, command packet length, response packet length, and time interval between two packets in the water storage dataset, as well as electrical pulses of junctions in the power grid dataset.

Due to the differences in scenarios, applications, communication protocols, data features, and classes of attacks of the above-mentioned datasets, it may take considerable efforts to adapt the proposed method for applying it to the datasets. However, we still plan to apply the proposed method to the above-mentioned datasets for verifying the effectiveness and the applicability of the method. In practice, we will first apply the proposed method to the water storage dataset [47], as the dataset adopts the Modbus protocol, which is also adopted by the Electra dataset, and has similar data features and

attack labels, such as response injection attacks and command injection attacks, to those of the Electra dataset.

Although the proposed method achieves the highest performance in ICS anomaly detection and very high performance in ICS anomaly classification, there are factors that may degrade its performance, especially for the anomaly classification case. One factor is that small class sizes may degrade the performance even if the oversampling mechanism is employed. We are planning to use few-shot meta learning mechanisms [49], [50] to mitigate the effect of this factor on performance. Another factor is that some anomalies or attacks may have very similar patterns in a single packet sample so that a class is misclassified as another class. For example, many read-attack samples in the Electra Modbus dataset are misclassified as replay-attack samples due to the fact that read-attack and replay-attack samples have similar patterns in network traffic, a fact already observed in [9]. Since many specific and contiguous packets usually precede a certain attack, it is useful to analyze a bunch of packets within a time window for detecting and classifying anomalies related to attacks. In the future, we plan to use recurrent neural network (RNN) [51] models, such as the long short-term memory (LSTM) neural network [52] and the gated recurrent unit (GRU) neural network [53] to extract feature from sequential packets to help distinguish attacks having similar patterns in a single packet sample. The negative effect of the second factor on performance may thus become milder. Moreover, we also plan to apply other potential methods to the Electra dataset for possible improvement of the multi-class classification performance. The potential methods are exemplified by the cost-sensitive decision tree, roughly balanced bagging, random oversampling bagging, random undersampling bagging, synthetic minority oversampling bagging, random undersampling boosting, and synthetic minority oversampling boosting methods that are investigated in [48].

We have noticed that a related paper [54] addressing the problem that machine learning models become ineffective when facing evasion attacks, a special type of adversarial attacks. Attackers launch evasion attacks by deliberately crafting fake data that are misclassified by machining learning anomaly detection methods for reaching industrial devices to disrupt ICS processes. Fortunately, the DAE can defend against adversarial attacks, as suggested by Mahfuz in [55]. The proposed method may somewhat resist to evasion attacks, since it integrates the DAE. In the future, we plan to investigate adversarial attacks and improve the proposed method so that it is less vulnerable to such attacks.

APPENDIX A
ILLUSTRATION OF A TEST DATUM GOING THROUGH DATA PREPROCESSING, DAE PROCESSING, AND XGBoost PROCESSING

The following shows a running example of a test data item from the Electra S7Comm dataset going through data preprocessing, DAE processing, and XGBoost processing.

An original data item from Electra S7Comm Dataset:

time	smac	dmac	sip	dip	req	fc	err	add	dat	lab
26457099029	08:00:27:79:b0:4a	28:63:36:a0:31:ea	10.70.38.131	10.70.38.52	1	5	0	11076	75	WRITE_ATTACK

Note that "time" stands for "timestamp", "smac" stands for "source MAC address", "dmac" stands for "destination MAC address", "sip" stands for "source IP address", "dip" stands for "destination IP address", "req" stands for "request", "fc" stands for "function code", "err" stands for "error", "add" stands for "memory address for operation", "dat" stands for "data for operation", and "lab" stands for "label of the data item".

After deleting the "time" and the "lab" features and one-hot encoding:

req	fc	err	add	dat	smac_ 00:1b: 1b:c1: 41:1b	smac_ 08:00: 27:79: b0:4a	smac_ 28:63: 36:a0: 31:ea	smac_ 28:63: 36:a2: 39:48	dmac_ 00:1b: 1b:c1: 41:1b	dmac_ 08:00: 27:79: b0:4a	dmac_ 28:63: 36:a0: 31:ea	dmac_ 28:63: 36:a2: 38:d7	dmac_ 28:63: 36:a2: 39:48	sip_10 .70.38 .131	sip_10 .70.38 .51	sip_10 .70.38 .52	sip_10 .70.38 .53	sip_10 .70.38 .54	dip_10 .70.38. 131	dip_10 .70.38. 51	dip_10 .70.38. 52	dip_10 .70.38. 53	dip_10 .70.38. 54
1	5	0	11076	75	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0

After label encoding:

req	fc	err	add	dat	smac_ 00:1b: 1b:c1: 41:1b	smac_ 08:00: 27:79: b0:4a	smac_ 28:63: 36:a0: 31:ea	smac_ 28:63: 36:a2: 39:48	dmac_ 00:1b: 1b:c1: 41:1b	dmac_ 08:00: 27:79: b0:4a	dmac_ 28:63: 36:a0: 31:ea	dmac_ 28:63: 36:a2: 38:d7	dmac_ 28:63: 36:a2: 39:48	sip_10 .70.38 .131	sip_10 .70.38 .51	sip_10 .70.38 .52	sip_10 .70.38 .53	sip_10 .70.38 .54	dip_10 .70.38. 131	dip_10 .70.38. 51	dip_10 .70.38. 52	dip_10 .70.38. 53	dip_10 .70.38. 54
1	1	0	3489	85	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0

After standard scaler processing:

req	fc	err	add	dat	smac_ 00:1b: 1b:c1: 41:1b	smac_ 08:00: 27:79: b0:4a	smac_ 28:63: 36:a0: 31:ea	smac_ 28:63: 36:a2: 39:48	dmac_ 00:1b: 1b:c1: 41:1b	dmac_ 08:00: 27:79: b0:4a	dmac_ 28:63: 36:a0: 31:ea	dmac_ 28:63: 36:a2: 38:d7	dmac_ 28:63: 36:a2: 39:48	sip_10 .70.38 .131	sip_10 .70.38 .51	sip_10 .70.38 .52	sip_10 .70.38 .53	sip_10 .70.38 .54	dip_10 .70.38. 131	dip_10 .70.38. 51	dip_10 .70.38. 52	dip_10 .70.38. 53	dip_10 .70.38. 54
1	0.871774	0	0.197020	-0.789013	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0

After DAE processing:

feature_1	feature_2	feature_3	feature_4	feature_5	feature_6	feature_7	feature_8	feature_9	feature_10	feature_11	feature_12
1.002787	0.6921923	-0.00665335	0.2350623	-0.7800894	0.01092414	0.9994013	0.00833078	0.00315288	0.00625425	0.00572486	1.0036371
feature_13	**feature_14**	**feature_15**	**feature_16**	**feature_17**	**feature_18**	**feature_19**	**feature_20**	**feature_21**	**feature_22**	**feature_23**	**feature_24**
0.00578448	0.00512569	0.96607363	0.01099516	0.01382149	0.00299251	0.0253805	0.00517722	0.00435427	1.0025623	0.00494824	0.00577043

After XGBoost processing:
Output the classification label -- WRITE_ATTACK

REFERENCES

[1] J.-R. Jiang, "An improved cyber-physical systems architecture for Industry 4.0 smart factories," *Adv. Mech. Eng.*, vol. 10, no. 6, 2018, Art. no. 1687814018784192.

[2] T. Yamada, T. Nakano, T. Kaji, and S. Tano, "Security introduction framework for operational technologies and applying to industrial control system," in *Proc. 59th Annu. Conf. Soc. Instrum. Control Eng. Jpn. (SICE)*, Sep. 2020, pp. 25–30.

[3] K. Stouffer, J. Falco, and K. Scarfone, "Guide to industrial control systems (ICS) security," *NIST Special Publication*, vol. 800, no. 82, p. 16, 2011.

[4] T. Alladi, V. Chamola, and S. Zeadally, "Industrial control systems: Cyberattack trends and countermeasures," *Comput. Commun.*, vol. 155, pp. 1–8, Apr. 2020.

[5] T. Miller, A. Staves, S. Maesschalck, M. Sturdee, and B. Green, "Looking back to look forward: Lessons learnt from cyber-attacks on industrial control systems," *Int. J. Crit. Infrastruct. Protection*, vol. 35, Dec. 2021, Art. no. 100464.

[6] S. Karnouskos, "Stuxnet worm impact on industrial cyber-physical system security," in *Proc. 37th Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, Nov. 2011, pp. 4490–4494.

[7] C. Zhou, B. Hu, Y. Shi, Y.-C. Tian, X. Li, and Y. Zhao, "A unified architectural approach for cyberattack-resilient industrial control systems," *Proc. IEEE*, vol. 109, no. 4, pp. 517–541, Apr. 2020.

[8] C. Eaton and D. Volz, "U.S. Pipeline cyberattack forces closure," *Wall Street J.*, May 2021. [Online]. Available: https://www.wsj.com/articles/cyberattack-forces-closure-of-largest-u-s-refined-fuel-pipeline-11620479737

[9] A. L. P. Gómez, L. F. Maimó, A. H. Celdran, F. J. G. Clemente, C. C. Sarmiento, C. J. D. C. Masa, and R. M. Nistal, "On the generation of anomaly detection datasets in industrial control systems," *IEEE Access*, vol. 7, pp. 177460–177473, 2019.

[10] A. R. B. Gupta and J. Agrawal, "A comprehensive survey on various machine learning methods used for intrusion detection system," in *Proc. IEEE 9th Int. Conf. Commun. Syst. Netw. Technol. (CSNT)*, Apr. 2020, pp. 282–289.

[11] *Open Modbus TCP Standard*. Accessed: Aug. 25, 2021. [Online]. Available: http://www.dankohn.info/projects/Fieldpoint_module/Open_ ModbusTCP_Standard.pdf

[12] A. Kleinmann and A. Wool, "Accurate modeling of the Siemens s7 SCADA protocol for intrusion detection and digital forensics," *J. Digit. Forensics, Secur. Law*, vol. 9, no. 2, p. 37, 2014.

[13] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. 25th Int. Conf. Mach. Learn. (ICML)*, 2008, pp. 1096–1103.

[14] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, no. 1, pp. 321–357, 2002.

[15] I. Tomek, "Two modifications of CNN," *IEEE Trans. Syst., Man Cybern.*, vol. 6, pp. 769–772, 1976.

[16] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 785–794.

[17] B. Ning, S. Qiu, T. Zhao, and Y. Li, "Power IoT attack samples generation and detection using generative adversarial networks," in *Proc. IEEE 4th Conf. Energy Internet Energy Syst. Integr. (EI)*, Oct. 2020, pp. 3721–3724.

[18] *Dataset for Cybersecurity Research in Industrial Control Systems*. Accessed: Apr. 30, 2021. [Online]. Available: http://perception.inf.um.es/ICS-datasets/

[19] T. H. Morris and W. Gao, "Industrial control system cyber attacks," in *Proc. Electron. Workshops Comput.*, Sep. 2013, pp. 22–29.

[20] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Commun. ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[21] K. Hara, D. Saito, and H. Shouno, "Analysis of function of rectified linear unit used in deep learning," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2015, pp. 1–8.

[22] E. Jackson and R. Agrawal, "Performance evaluation of different feature encoding schemes on cybersecurity logs," in *Proc. SoutheastCon*, Apr. 2019, pp. 1–9.

[23] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," Jolla Inst. Cogn. Sci., California Univ., San Diego, LA, USA, ICS Rep. 8506, 1985.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[25] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Statist.*, vol. 29, no. 5, pp. 1189–1232, Oct. 2001.

[26] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Evanston, IL, USA: Routledge, 2017.

[27] X. Luo, X. Chang, and X. Ban, "Regression and classification using extreme learning machine based on L_1-norm and L_2-norm," *Neurocomputing*, vol. 174, pp. 179–186, Jan. 2016.

[28] S. Haykin, *Neural Networks and Learning Machines, 3/E*. London, U.K.: Pearson, 2010.

[29] X. Xu, W. Chen, and Y. Sun, "Over-sampling algorithm for imbalanced data classification," *J. Syst. Eng. Electron.*, vol. 30, no. 6, pp. 1182–1191, 2019.

[30] S. Sridhar and S. Sanagavarapu, "Handling data imbalance in predictive maintenance for machines using SMOTE-based oversampling," in *Proc. 13th Int. Conf. Comput. Intell. Commun. Netw. (CICN)*, Sep. 2021, pp. 44–49.

[31] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[32] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," 2019, *arXiv:1904.09237*.

[33] F. Zou, L. Shen, Z. Jie, W. Zhang, and W. Liu, "A sufficient condition for convergences of adam and RMSProp," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 11127–11135.

[34] M. Zaheer, S. Reddi, D. Sachan, S. Kale, and S. Kumar, "Adaptive methods for nonconvex optimization," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst. (NIPS)*, Montréal, QC, Canada, Dec. 2018, pp. 9815–9825.

[35] A. Défossez, L. Bottou, F. Bach, and N. Usunier, "A simple convergence proof of adam and adagrad," 2020, *arXiv:2003.02395*.

[36] N. V. Chawla, "Data mining for imbalanced datasets: An overview," in *Data Mining and Knowledge Discovery Handbook*. New York, NY, USA: Springer, 2009, pp. 875–886.

[37] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 6765–6816, 2017.

[38] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 972–981.

[39] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, 2010, pp. 249–256.

[40] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[41] T.-H. Lin and J.-R. Jiang, "Credit card fraud detection with autoencoder and probabilistic random forest," *Mathematics*, vol. 9, no. 21, p. 2683, Oct. 2021.

[42] A. P. Mathur and N. O. Tippenhauer, "SWaT: A water treatment testbed for research and training on ICS security," in *Proc. Int. Workshop Cyber-Phys. Syst. Smart Water Netw. (CySWater)*, Apr. 2016, pp. 31–36.

[43] C. M. Ahmed, V. R. Palleti, and A. P. Mathur, "WADI: A water distribution testbed for research in the design of secure cyber physical systems," in *Proc. 3rd Int. Workshop Cyber-Phys. Syst. Smart Water Netw.*, Apr. 2017, pp. 25–28.

[44] S. Adepu, N. K. Kandasamy, and A. Mathur, "Epic: An electric power testbed for research and training in cyber physical systems security," in *Computer Security*. Cham, Switzerland: Springer, 2018, pp. 37–52.

[45] J. M. Beaver, R. C. Borges-Hink, and M. A. Buckner, "An evaluation of machine learning methods to detect malicious SCADA communications," in *Proc. 12th Int. Conf. Mach. Learn. Appl. (ICMLA)*, vol. 2, Dec. 2013, pp. 54–59.

[46] S. Pan, T. Morris, and U. Adhikari, "Developing a hybrid intrusion detection system using data mining for power systems," *IEEE Trans. Smart Grid*, vol. 6, no. 6, pp. 3104–3113, Nov. 2015.

[47] T. Morris, A. Srivastava, B. Reaves, W. Gao, K. Pavurapu, and R. Reddi, "A control system testbed to validate critical infrastructure protection concepts," *Int. J. Crit. Infrastruct. Protection*, vol. 4, no. 2, pp. 88–103, 2011.

[48] M. H. L. Louk and B. A. Tama, "Exploring ensemble-based class imbalance learners for intrusion detection in industrial control networks," *Big Data Cognit. Comput.*, vol. 5, no. 4, p. 72, Dec. 2021.

[49] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. D. Freitas, "Learning to learn by gradient descent by gradient descent," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3981–3989.

[50] C. Xu, J. Shen, and X. Du, "A method of few-shot network intrusion detection based on meta-learning framework," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3540–3552, 2020.

[51] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, Oct. 1986.
[52] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[53] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," 2014, *arXiv:1406.1078*.
[54] Á. L. P. Gómez, L. F. Maimó, A. H. Celdrán, F. J. G. Clemente, and F. Cleary, "Crafting adversarial samples for anomaly detectors in industrial control systems," *Proc. Comput. Sci.*, vol. 184, pp. 573–580, Jan. 2021.
[55] R. Mahfuz, "Defending against adversarial attacks using denoising autoencoders," Ph.D. dissertation, Purdue Univ. Graduate School, West Lafayette, IN, USA, 2020.

JEHN-RUEY JIANG (Member, IEEE) received the Ph.D. degree in computer science from the National Tsing Hua University, Hsinchu, Taiwan, in 1995. In 1995, he joined Chung Yuan Christian University as an Associate Professor. In 1998, he joined Hsuan Chuang University and became a Full Professor, in 2004. He is currently with the Department of Computer Science and Information Engineering, National Central University, Taoyuan, Taiwan. He also leads the Advanced Computing and Networking (ACAN) Laboratory, which focuses on investigating advanced technologies about networking and computing. His research interests include the Internet of Things (IoT), machine learning/deep learning technologies, smart manufacturing, cyber-physical systems for industry 4.0, and quantum computing.

YAN-TING CHEN received the B.S. degree from the Department of Finance, Feng Chia University, Taichung, Taiwan, in 2017, and the M.S. degree from the Department of Computer Science and Information Engineering, National Central University, Taoyuan, Taiwan, in 2021. His research interests include big data analysis and machine learning/deep learning.