IEEE Open Journal of Instrumentation and Measurement

Received 21 May 2024; accepted 10 June 2024. Date of publication 24 June 2024; date of current version 19 July 2024.
The review of this paper was arranged by Associate Editor Jacob Scharcanski.

Digital Object Identifier 10.1109/OJIM.2024.3418466

Multigranularity Feature Automatic Marking-Based Deep Learning for Anomaly Detection of Industrial Control Systems

XINYI DU [1,2,3], CHI XU [2,3] (Senior Member, IEEE), LIN LI [2], AND XINCHUN LI [1]

[1]School of Electronic and Information Engineering, Liaoning Technical University, Huludao 125105, China

[2]State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China

[3]Key Laboratory of Networked Control Systems, Chinese Academy of Sciences, Shenyang 110016, China

CORRESPONDING AUTHOR: C. XU (e-mail: xuchi@sia.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 92267108 and Grant 62173322, and in part by the Science and Technology Program of Liaoning Province under Grant 2023JH3/10200004 and Grant 2022JH25/10100005.

(Xinyi Du and Chi Xu contributed equally to this work.)

ABSTRACT Industrial control systems are facing ever-increasing security challenges due to the large-scale access of heterogeneous devices in the open Internet environment. Existing anomaly detection methods are mainly based on the priori knowledge of industrial control protocols (ICPs) whose protocol specifications, communication mechanism, and data format are already known. However, when these knowledge are blank, namely, unknown ICPs, existing methods become powerless to detect the anomaly data. To tackle this challenge, we propose a multigranularity feature automatic marking-based deep learning method to classify unknown ICPs for anomaly detection. First, to obtain the feature sequences without priori knowledge assisting, we propose a multigranularity feature extraction algorithm to extract both byte and half-byte information by fully utilizing the intensive key information in the header field of the application layer. Then, to label the feature sequences for deep learning, we propose a feature automatic marking algorithm that utilizes the inconsistency feature sequences to dynamically update the feature sequence set. With the labeled feature sequences, we employ deep learning with 1-D convolutional neural network and gated recurrent unit to classify the unknown ICPs and realize anomaly detection. Extensive experiments on two public datasets show that both the accuracy and precision of the proposed method reach above 98.4%, which is better than the three benchmark methods.

INDEX TERMS Anomaly detection, convolutional neural network, deep learning, feature automatic marking, feature extraction, industrial control protocol (ICP).

I. INTRODUCTION

WITH the vigorous development of industrial automation and informatization, industrial control systems (ICSs) play an increasingly critical and indispensable role in the production process [1], [2]. However, ICSs are also facing ever-increasing security challenges due to the large-scale access of heterogeneous devices in the open Internet environment, where network attacks, malicious software infections, and physical intrusions all have the potential to cause serious damage to ICSs, leading to production interruptions, equipment damage, and even safety accidents. Thus, anomaly detection technology becomes an important means to ensure the security of ICSs. Through anomaly detection, abnormal behavior or faults in ICSs can be monitored and identified, enabling the timely detection of potential security threats and the implementation of necessary measures. Thus, anomaly detection not only contributes to enhancing the security of ICS itself but also mitigates potential impacts on the entire production and society [3].

Most existing anomaly detection methods are performed based on known industrial control protocols (ICPs) with

priori knowledge, such as protocol specifications, communication mechanism, and data format. In this way, we can determine whether there are abnormal behaviors by learning and comparing the features with existing rules or behavior models. However, different ICS manufacturers develop numerous public and private ICPs, such as Modbus, Profibus, and CAN [4]. Each of these ICPs has its own data format, communication method, and instruction structure, requiring the anomaly detection methods to possess and understand multiple ICPs. In real industrial environments, it is often the case that multiple different ICPs are used simultaneously, making the anomaly detection more challenging. Meanwhile, ICSs are constantly updated and evolved with new ICPs, such as WIA and 5G [5]. This means that some ICPs may not be included in the predefined detection scope and cannot be detected without priori knowledge. More importantly, many ICS manufacturers do not open their ICP specifications to protect security and privacy. As a consequence, existing anomaly detection methods for known ICPs become invalid for unknown ICPs when there is no priori knowledge. To detect unknown ICPs, it is necessary to first classify and recognize unknown ICPs, assigning them to specific known protocol types, and then establish behavioral models to perform targeted anomaly detection, providing more accurate and effective feature information for the detection process.

Protocol reverse engineering is an important approach to analyze different kinds of protocols. Narayan et al. [6], Huang et al. [7], and Kleber et al. [8] have provided full overviews on the protocol reverse engineering from different perspective, including protocols feature analysis, protocol classification, tool development, and so on. Obviously, by employing the reverse thinking, one can infer the syntax, semantics, and time sequences of a protocol, thus enabling the identification of unknown ICPs. Generally, the methods for protocol reverse analysis primarily rely on program instructions or message sequences. The program instruction-based reverse methods have strong analytical capabilities and yield accurate results. However, it is challenging to obtain the source code of the ICPs program. In contrast, the message sequence-based reverse methods are relatively easier to acquire the protocol, and they can handle similar protocols.

Previously, Wang and Li [9] have developed an automatic protocol reverse engineering tool called IPART based on global voting expert for ICPs. Marchetti and Stabili [10] have proposed a novel algorithm named READ for the automatic reverse engineering of automotive data frames over CAN in-vehicle networks as the public specifications for CAN messages are lacking. More recently, Ning et al. [11] have presented an ICP reverse engineering tool named PREIUD which is based on unsupervised learning and deep neural network. However, most existing methods directly detect the anomaly data by reverse engineering without classifying the known and unknown ICPs, which limits the accuracy. Motivated by this, this article employs the basic idea of protocol reverse engineering and proposes multigranularity feature automatic marking-based deep learning (MGFAM-DL) method to classify unknown ICPs for anomaly detection, where multigranularity feature extraction and automatic marking algorithm is proposed to facilitate deep learning and enhance the accuracy. The main contributions of this article are as follows.

1) To break the feature reliance and subjectivity of existing algorithms for known ICPs and support unknown ICPs, we take into account the intensive key information in the header field of the application layer (APL) and propose a multigranularity feature extraction algorithm combining byte and half-byte to construct a feature sequence set for feature marking.
2) To mark the feature sequences of unknown ICPs and enhance the efficiency for deep learning, we propose a feature automatic marking algorithm by considering the inconsistency of feature sequences in the APL header-field sequences. By evaluating feature variations, the algorithm can automatically mark APL header-field sequences with the same feature byte sequences and dynamically adjust the labels, solving the time consuming and inaccuracy and inefficient problem of manual marking.
3) To learn the temporal and spatial features of the marked APL header-field sequences, we combine 1-D convolutional neural network (1D-CNN) and gated recurrent unit (GRU) and propose the deep-learning-based anomaly detection method by fully considering the limited computation resources of industrial equipments and the time-series features of ICPs.

The remainder of this article is organized as follows. Section II presents the related works. Section III demonstrates the multigranularity feature extraction and automatic marking process. Section IV proposes the deep learning algorithm for anomaly detection. Section V presents experimental results. Finally, the work is concluded in Section VI.

For easy reading, the abbreviations are summarized in Table 1.

TABLE 1. Abbreviation summary.

Abbreviation	Full explanation
ICS	industrial control system
ICP	industrial control protocol
APL	application layer
DL	deep learning
GRU	gate recurrent unit
CNN	convolutional neural network
1D-CNN	one-dimensional convolutional neural network
LSTM	long short-term memory network
MGFAM-DL	multi-granularity feature automatic marking DL
AM-1D-CNN+LSTM	attention-enhanced 1D-CNN and LSTM
TP	true positive
FP	false positive
FN	false negative
TN	true negative

TABLE 2. Method comparison.

Reference	Year	Protocol	Algorithm	Algorithm Category	Objective
Ref. [12]	2021	Known	DL	Supervised	Intrusion detection
Ref. [13]	2021	Known	DL+SVM	Supervised	Traffic detection
Ref. [14]	2022	Known	Decision tree	Supervised	Feature extraction
Ref. [15]	2022	Known	DL+XGBoost	Supervised	Anomaly detection and protocol classification
Ref. [16]	2023	Known	Density clustering	Unsupervised	Protocol pre-classification
Ref. [17]	2023	Known	DL	Supervised	Anomaly detection and protocol classification
Ref. [18]	2020	Unknown	DL	Semi-supervised	protocol classification
Ref. [9]	2020	Unknown	Global voting expert	Unsupervised	Protocol classification
Ref. [19]	2021	Unknown	DL	Supervised	Protocol identification
Ref. [20]	2021	Unknown	DL	Supervised	Protocol classification
Ref. [21]	2022	Unknown	Eps-neighborhood	Semi-supervised	Protocol identification

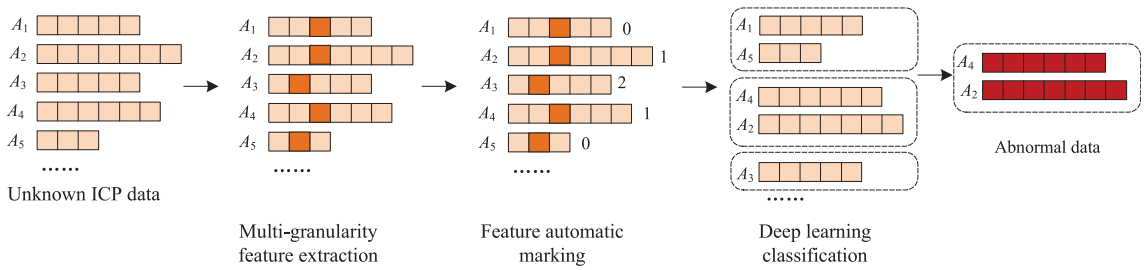


FIGURE 1. Overall framework of the method.

II. RELATED WORK

The multitude of protocol types, complex protocol formats, and changeable attack methods contribute to the increasing complexity of methods. In recent years, researchers extensively explored the ICP classification methods for anomaly detection, including both unsupervised learning and supervised learning for both known and unknown ICPs, as shown in Table 2.

Ling et al. [12] proposed a method for detecting anomalies and classifying large-scale network traffic data in ICSs based on bidirectional simple recurrent units. Sestito et al. [13] proposed to use artificial neural network and support vector machines to classify industrial Ethernet protocols. Yu et al. [14] proposed a method to extract the keyword features of ICPs and classify them with decision trees. Jiang and Chen [15] proposed method uses the denoising autoencoder, synthetic minority oversampling technique, T-Link, and XGBoost mechanisms to achieve multiclass classification of abnormal ICPs. Jin et al. [16] proposed an adaptive density clustering method for preclassification ICPs. Zhang et al. [17] proposed a neural network structure to classify attacks on Modbus protocol at a fine-grained level. By conducting an analysis of various ICPs, researchers can categorize different types of known ICPs, providing crucial support for the development of ICS. However, the research on the classification of unknown ICPs is at an early stage.

Several studies explore different approaches for unknown protocol classification. For example, Zhao and Liu [18] used a public protocol datasets to train a long short-term memory fully convolutional neural network for classifying private ICPs. Wang and Li [9] proposed an unsupervised tool for automatically reversing the format of the industrial protocol from network trace. Zhai et al. [19] preprocessed the effective payload of ICPs and accomplished classification through a fusion of long short-term memory network (LSTM) with added attention mechanism and 1D-CNN. Lin et al. [20] introduced stacked bidirectional LSTMs into convolutional neural networks for encryption protocol classification. Zhu et al. [21] utilized the distinctive feature of known protocols, one can classify unknown protocols after dimension reduction using the neighborhood-hit method. The above studies demonstrate the efforts of deep learning for unknown ICP classification. Hence, recognizing the immense potential of deep learning in anomaly detection for unknown ICPs, this article proposes a deep learning method for classifying unknown ICPs.

III. MULTIGRANULARITY FEATURE EXTRACTION AND AUTOMATIC MARKING

As depicted in Fig. 1, the proposed MGFAM-DL method mainly includes two steps: 1) feature extraction and marking and 2) deep learning. Obviously, feature extraction and marking are the precondition for deep learning. Thus, this section introduces the multigranularity feature extraction and automatic marking algorithm.

A. MULTIGRANULARITY FEATURE EXTRACTION

To extract the features of unknown ICPs, we make full use of the intensive key information in the APL header field and propose the multigranularity feature extraction

algorithm, which mainly includes four steps: 1) sequence set construction; 2) sequential byte comparison; 3) interchanging half-byte comparison; and 4) feature byte sequences filter.

1) SEQUENCE SET CONSTRUCTION

First, we convert the raw binary numbers from M ICPs data to hexadecimal numbers for easy extraction of protocol information. For example, the binary number 10000001 is converted to the hexadecimal number 0x81, representing the protocol identification of the BACnet protocol. Next, we extract the key information from the APL header-field sequences of ICPs data, including protocol type, version, and device address. Note that different ICPs may have varying APL header-field sequence lengths. Thus, in order to avoid the high overhead and delay issues caused by redundant features, we design a baseline length to extract ICPs data to ensure comparability.

In particular, we calculate the length of all ICPs data one by one and select the shortest length as the baseline length. Then, we extract the APL header-field sequences equal to the baseline length, starting from the first byte. In this way, we construct an APL header-field sequence set $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_M)$, where σ_m denotes the mth APL header-field sequence. After that, we select any two APL header-field sequences from σ to create a binary tuple. The set of all binary tuples is denoted as $\zeta = \{(\sigma_m, \sigma_n) \mid m, n = 1, 2, \ldots, M, m \neq n\}$, where (σ_m, σ_n) is a binary tuple for comparison in the next step.

2) SEQUENTIAL BYTE COMPARISON

With binary tuple sequences, considering the differential feature of byte at corresponding positions in different types of ICP, we compare the binary tuple sequences byte-by-byte where each byte is regarded as a unit.

Specifically, we first calculate the byte length of binary tuple sequences. As each eight-bit binary number can be converted into a two-bit hexadecimal number, representing one byte, we calculate the byte length as σ_m is $H = \dfrac{h}{8}$, where h represents the length of the binary tuple sequences. Then, we calculate the byte comparison score. The differences between the byte σ_{mi} and σ_{ni} at the same position in σ_m and σ_n from left to right are compared and the score for the ith comparison is calculated as

$$S(i) = \begin{cases} 1, \sigma_{mi} = \sigma_{ni} \\ 0, \sigma_{mi} = \sigma_{ni} \end{cases}, i = 1, \ldots, H. \tag{1}$$

Next, the score from each comparison is accumulated to obtain the total score, denoted as $S_{\text{total}} = \sum_{i=1}^{H} S(i)$, for the comparison of a binary tuple sequence. In order to eliminate the impact of binary tuple sequences length on the score, the average of S_{total} denoted as $S_{\text{aver}} = (S_{\text{total}}/H)$. The byte score matrix of binary tuple sequences is denoted as $\mathbf{D}_1$.

3) INTERCHANGING HALF-BYTE COMPARISON

However, the byte order comparison method can only determine the correlation of binary tuple sequences at a coarse-grained level. For similar binary tuple sequences with small differences, as most of their bytes are identical, the S_{aver} fails to accurately reflect their correlation. Therefore, we further propose a half-byte transposition comparison method based on the Jaro–Winkler method [22] to evaluate the correlation of binary tuple sequences at a fine-grained level.

Specifically, we first calculate the half-byte length of binary tuple sequences. As each half-byte consists of four binary bits, the half-byte length of binary tuple sequences is calculated as $H_1 = (h/4)$. Then, we calculate the matching window size. To prevent excessive time and computational resources consumed when comparing long binary tuple sequences, we set a dynamic range, namely, matching window $L = (H_1/2) - 1$. By this, the position distance of the two half-bytes being compared must not exceed the size of the matching window. If the distance exceeds the matching window size, those half-bytes are considered as nonmatching. Next, we calculate the half-byte comparison score. We traverse each half-byte σ_{mj} in σ_m from left to right and find a matching half-byte σ_{nj} in σ_n within the matching window. If $\sigma_{mj} = \sigma_{nj}$, σ_{mj} and σ_{nj} are regarded as matching half-bytes and the number of matching half-bytes within the matching window range is counted, denoted as R. Finally, we extract the matching half-bytes from their binary tuple sequences to form a new binary tuple sequence set $\tau = \{(\tau_m, \tau_n) \mid m, n = 1, 2, \ldots, M, m \neq n\}$, where (τ_m, τ_n) is an ordered binary tuple sequences. If $\tau_{mj} \neq \tau_{nj}$, there is one-time transposition, and the number of half-bytes involved in the transposition is half of the number of transposition time, denoted as T.

Through the above process, we can calculate the half-byte comparison score as

$$S_{\text{h-total}} = S_{\text{h-ptotal}} + 0.1p\left(1 - S_{\text{h-ptotal}}\right) \tag{2}$$

where $S_{\text{h-ptotal}} = (1/2)([R/H_1] + [R - T/R])$ represents the half-byte comparison score without considering the length of binary tuple sequences prefix. p refers to the number of same bytes in the starting portion of binary tuple sequences, also known as the prefix length of binary tuple sequences. The half-byte score matrix formed by $S_{\text{h-total}}$ is denoted as $\mathbf{D}_2$.

4) FEATURE BYTE SEQUENCES FILTER

In order to select representative and distinctive sequences as the basis for feature automatic marking, we integrate $\mathbf{D}_1$ and $\mathbf{D}_2$ as

$$\mathbf{D}_3 = \frac{\mathbf{D}_1 + \mathbf{D}_2}{2} = \frac{1}{2}\begin{pmatrix} S_1 + S_{h-1} \\ S_2 + S_{h-2} \\ \vdots \\ S_M + S_{h-M} \end{pmatrix} \tag{3}$$

which reflects the correlation of binary tuple sequences. In $\mathbf{D}_3$, the numerical value of each element ranges from 0 to 1, where 0 represents the least related and 1 represents the most related.

We set the score threshold as q and retain the binary tuple sequences that exceed this threshold while those not exceed are discarded. Utilizing the $\mathbf{D}_3$ filter, we extract highly correlated binary tuple sequences to obtain σ_m and σ_n. If $\sigma_{mi} = \sigma_{ni}$, retain the byte; otherwise, use "–" as a placeholder for inconsistencies.

By repeating the above process until each binary tuple sequence contains only one byte, we obtain the feature byte sequences set $\nu = (\nu_1, \nu_2, \ldots, \nu_M)$, where ν_m represents the mth feature byte sequence.

B. FEATURE AUTOMATIC MARKING

With the extracted feature byte sequences, we assign labels to them. While manual marking can be chosen when the data volume is small, ICP data is often large and needs to meet real-time requirements. Therefore, we propose an automatic feature marking algorithm to address the time-consuming, labor-intensive, and error-prone nature of manual marking. It utilizes the inconsistency of feature byte in different ICPs.

1) FEATURE BYTE SEQUENCE SET COPY

To expedite the marking process, the marked feature byte sequences are filtered out from ν. In order to preserve the feature byte sequence, each element of ν is copied to formulate a new feature byte sequence set $\hat{\nu} = (\hat{\nu}_1, \hat{\nu}_2, \ldots, \hat{\nu}_M)$, where ν_m represents the mth feature byte sequence. The main reason is that the order of different feature byte sequences in ν is unordered. If the feature byte sequences in ν are deleted and the next round of filtering is carried out, there is a possibility of missing out on some feature byte sequences, leading to erroneous label results.

2) SEQUENCE AUTOMATIC MARKING

In order to enhance the accuracy of the label results, the feature byte sequences in ν and $\hat{\nu}$ are analyzed individually. A feature byte sequence $\hat{\nu}_m$ is automatically extracted from $\hat{\nu}$ and compared against all feature byte sequences in ν. According to the comparison results, it is recorded as a label, resulting in the mth label $G(m)$ as follows:

$$G(m) = \begin{cases} t, \nu_m = \hat{\nu}_m \\ r, \nu_m \neq \hat{\nu}_m \end{cases}, m = 1, \ldots, M \tag{4}$$

where t is a non-negative integer value and denotes the labels assigned to the same feature byte sequences of APL header-field sequences, and r denotes a placeholder label. When all feature byte sequences in ν and $\hat{\nu}$ are compared, $\hat{\nu}_m$ is filtered out of $\hat{\nu}$.

Repeating the above process until all the feature byte sequences in $\hat{\nu}$ are filtered out, we complete the APL header-field sequences marking. The label matrix $\mathbf{Q}$ is given as

Algorithm 1 Multigranularity Feature Automatic Marking

Input: $\sigma = (\sigma_1, \sigma_2, ..., \sigma_M)$;
Output: $\mathbf{Q}_1$;
for *comparing rounds* $= 1, 2, ...$ **do**
  Construct binary tuple sequences set $\zeta = \{(\sigma_m, \sigma_n) \mid m, n = 1, 2, ..., M, m \neq n\}$;
  Calculate byte length H of σ_m and σ_n;
  Compare σ_{mi} and σ_{ni} by (1);
  Calculate byte comparison score S_{total};
  Average S_{total} to S_{aver};
  Obtain byte score matrix $\mathbf{D}_1$;
  Calculate half-byte length H_1 of σ_m and σ_n;
  Calculate matching window size L;
  Compare σ_{mj} and σ_{nj};
  Calculate half-byte comparison score $S_{\text{h-total}}$ by (2);
  Obtain half-byte score matrix $\mathbf{D}_2$;
Integrate $\mathbf{D}_1$ and $\mathbf{D}_2$ to obtain $\mathbf{D}_3$ by (3);
Obtain feature byte sequences $\nu = (\nu_1, \nu_2, ..., \nu_M)$;
Copy ν to $\hat{\nu}$;
for *comparing rounds* $= 1, 2, ...$ **do**
  Compare ν_m and $\hat{\nu}_m$ by (4);
  Filter $\hat{\nu}_m$ from $\hat{\nu}$;
  $\hat{\nu}$ is empty;
Obtain label matrix $\mathbf{Q}$ as (5);
Dynamically adjust label r;
Obtain final label matrix $\mathbf{Q}_1$ as (6);

$$\mathbf{Q} = \begin{pmatrix} 0 & r & \cdots & r \\ r & 1 & \cdots & r \\ r & 1 & \cdots & r \\ \vdots & \vdots & \ddots & \vdots \\ r & r & \cdots & t \end{pmatrix}. \tag{5}$$

3) DYNAMICALLY ADJUST LABEL MATRIX

After multiple iterations, the APL header-field sequences of the ICPs are marked with label t and placeholder label r simultaneously. The label r is used to distinguish t, essentially serving as an invalid label. In order to retain a valid label, we dynamically adjust the label in the label matrix by calculating the frequency of appearance of each σ_m corresponding label, filtering out r, and obtaining the final label matrix as

$$\mathbf{Q}_1 = [\ 0\ 1\ 1 \cdots\ t\ ]^{\mathbf{T}}. \tag{6}$$

Algorithm 1 summarizes the multigranularity feature extraction and automatic marking process proposed in this section, denoting as multigranularity feature automatic marking. With the marked APL header-field sequences, we can employ deep learning to classify the unknown ICPs for anomaly detection.

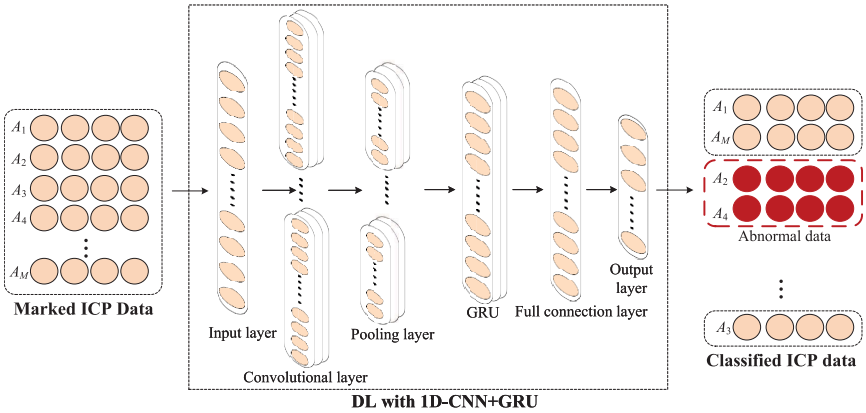


FIGURE 2. Deep learning structure combining 1D-CNN and GRU.

IV. DEEP LEARNING FOR ANOMALY DETECTION

With the automatically marked features, we further propose a deep learning algorithm to classify the ICPs for anomaly detection. Due to the limited computational resources of industrial equipments in ICSs, it is imperative to craft a simplistic network structure. As ICPs typically exhibit intricate spatial features and temporal correlations, we employ 1D-CNN to capture spatial features and combine with GRU to capture temporal features. By incorporating GRU on top of 1D-CNN, it enables rapid and effective capturing of spatial features and temporal dependencies in the data, thereby further improving classification performance. Besides, by leveraging the hierarchical understanding of data features through a stacked architecture, the network's overall performance and generalization capabilities are enhanced. The overall structure of the proposed deep learning algorithm is given in Fig. 2.

A. DATA PREPROCESSING

As the discrete byte sequences in the ICP headers cannot be directly used for training, we propose to adopt label encoding to numerically represent the ICP header sequences, thereby enhancing the feature expression capability. Furthermore, the processed data is normalized to improve the iteration speed of the method and obtain an optimal solution through gradient descent. Herein, Min–Max normalization is used to linearly transform the data, ensuring that the results fall within the range of [0, 1].

B. SPATIAL FEATURE CAPTURING

To guarantee the learning capability of 1D-CNN, we employ X convolutional layers to strike a balance between method accuracy and real-time performance. These layers utilize the convolution operation between the convolutional kernel and input data to extract ICP features. The convolution operation is given as

$$x^{\text{conv}} = f\left(x^{\text{conv}-1} + A_k^{\text{conv}} + B_k^{\text{conv}}\right), 1 \le k \le K \tag{7}$$

where x^{conv} and $x^{\text{conv}-1}$ represent the output and input features of the convolutional layer, respectively; A_k^{conv} and B_k^{conv} denote the weight and bias factors of the convolutional layer; $*$ operator denotes the convolution operation, K denotes the total number of convolutional kernels, and $f(\cdot)$ denotes the activation function.

Furthermore, in order to enhance the data processing capabilities, a rectified linear function is employed for nonlinear transformation. By this, we can constrain the output values within a fixed range that can improve the capability of dealing with complex ICP data. The calculation formula is given as

$$y^{\text{conv}} = \text{ReLU}(x) = \max\left(x^{\text{conv}}, 0\right) \tag{8}$$

where $\text{ReLU}(\cdot)$ is the activation function.

C. DIMENSION REDUCTION

To reduce data processing and improve computational efficiency, a pooling layer is added after the convolutional layer to perform dimensionality reduction and reduce network parameters. Specifically, the maximum pooling is employed to highlight important features in ICP, avoiding the excessive smoothing of data caused by average pooling and preventing the loss of critical features. The calculation formula is given as

$$x^{\text{max pool}} = \max\left(y^{\text{conv}}\right) \tag{9}$$

where y^{conv} is the transformed input after activation, and $x^{\text{max pool}}$ is the output after the maximum pooling.

On this basis, the convolutional layer and the pooling layer are alternately stacked to continuously extract features while reducing the data dimension. Specifically, Y pooling layers are added between X convolutional layers, gradually decreasing the dimension of the input data while preserving key feature information, thus enhancing the computational efficiency.

D. TEMPORAL FEATURE CAPTURING

To enhance the nonlinear learning capability and memory capacity of 1D-CNN, we employ Z stacked GRU networks to strike a balance between the depth and efficiency of the method. In this way, we also accelerate convergence speed and training efficiency.

TABLE 3. Hyperparameters setup.

Parameter	Batch Size	Learning Rate	Iteration Number	Convolution Kernel Size
Value	512	0.001	20	4

E. OPTIMIZATION

To avoid overfitting in the iterative process during training, we further add a random dropout layer in the regularization layer to reduce overfitting. One instance is set inside the GRU layer to randomly pause the hidden state. Another instance is set between fully connected layers to increase the network's generalization capability. By setting the dropout rate in the random dropout layer, a certain proportion of neuron nodes in the network are frozen, and temporarily excluded from data information computation, thereby enhancing the robustness of the method.

F. COMPUTATIONAL COMPLEXITY

Through feature extraction and marking, and deep learning, we can classify the unknown ICPs for anomaly detection. As the feature extraction and marking are executed with linear complexity, the total computational complexity of the proposed MGFAM-DL method is mainly based on that of deep learning, including convolutional layer, pooling layer, GRU, and full connection layer. Specifically, the complexity of the convolutional layer is $\mathcal{O}(X \times K \times d \times x^{\text{conv}-1} \times x^{\text{conv}})$, where there are K kernels with size d. The complexity of the pooling layer is $\mathcal{O}(Y \times p \times y^{\text{conv}})$, where p is the pooling size. The complexity of GRU is $\mathcal{O}(Z \times x^{\text{max pool}} \times h^2)$, where Z is the number of layers with hidden state size h. The complexity of the full connection layer is $\mathcal{O}(G \times F)$, where F is the number of neurons and G is the number of outputs. In this way, the total computational complexity is $\mathcal{O}(I \times (X \times K \times d \times x^{\text{conv}-1} \times x^{\text{conv}} + Y \times p \times y^{\text{conv}} + Z \times x^{\text{max pool}} \times h^2 + G \times F))$, where I denotes the number of iterations.

V. EXPERIMENTAL ANALYSIS

A. EXPERIMENTAL SETUP

The proposed method is evaluated based on the Python Keras framework running on Intel Core i7-11700@3.60-GHz CPU, and 32-GB memory. The configuration of hyperparameters is presented in Table 3. Specifically, due to the large and complex nature of the ICP dataset, we select a large batch size that can effectively utilize computational resources for training. To help the model converge more stably during training, we set a small learning rate. To prevent overfitting while ensuring the model adequately learns the data, the number of iterations is chosen based on the results of each training session. As the size of convolutional kernels affects the range of feature extraction in convolution operations, we set a small kernel that is suitable for extracting local features present in ICP data. Furthermore, the proposed method is optimized using a simple and efficient adaptive moment estimation algorithm. The decay rate for the calculation of the first moment estimation is set to 0.95, implying a high contribution of historical gradients and a less impact from current gradients. This adjustment allows for smooth convergence and reduces oscillation during the training process.

We select two datasets for experiments, namely, the ICS village at 4SICS (Dataset 1) and Digital Bond S4 *15 ICS Village CTF PCAPS (Dataset 2). The datasets are obtained from the Netresec sharing website, which provides real-world industrial control business data and simulated data from large-scale experiments [23]. These datasets are collected during the same period, thus exhibiting strong spatiotemporal correlations, making them suitable for evaluating the proposed method.

Furthermore, we perform two kinds of comparative experiments to validate the effectiveness and superiorities of the proposed method on the two datasets. The first comparative experiment aims to eliminate different components of the proposed method and analyze their contributions. Thus, 1D-CNN and GRU are separately eliminated and compared with the proposed method. The second comparative experiment compares the proposed method with the attention-enhanced 1D-CNN and LSTM method (AM-1D-CNN+LSTM) similar to [19]. To evaluate and compare the performance of different methods, we evaluate four metrics, namely, Accuracy, Precision, Recall, and Score, as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{10}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{11}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{12}$$

$$\text{Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{13}$$

Among them, *TP* is the number of correctly predicted categories when predicting the true category, *TN* is the number of correctly predicted categories when predicting the false category, *FP* is the number of wrongly predicted categories when predicting the false category, and *FN* is the number of wrongly predicted categories when predicting the true category.

B. RESULTS ANALYSIS

Fig. 3 provides two confusion matrices, whose diagonal elements indicate the number of times the proposed method correctly detects a category, while the off-diagonal elements indicate the number of misdetection. In Fig. 3(a) for Dataset 1, Category 2 ICP is the most susceptible to be misdetected, while Category 3 ICP is the most easy to be detected correctly. In contrast, as depicted in Fig. 3(b) for Dataset 2, Category 3 ICP is the most susceptible to be misdetected, while Category 4 ICP is the most easy to be detected correctly. This is because there are indistinguishable features among different categories. That is to say, Category 2 ICP in Dataset 1 and Category 3 ICP in Dataset 2 are highly similar

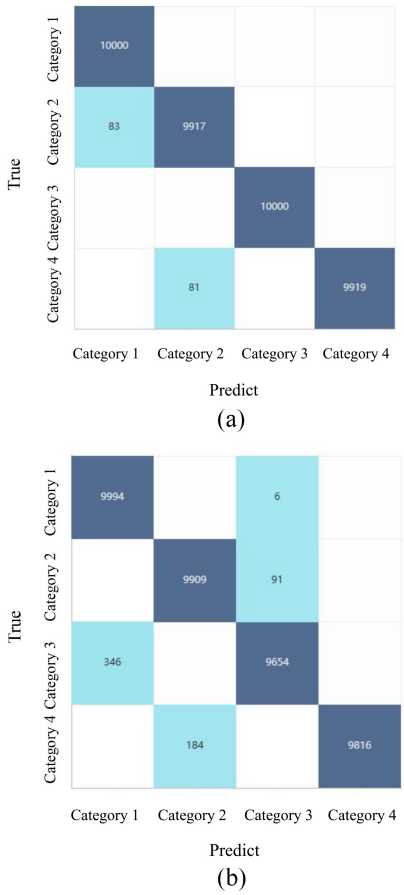


FIGURE 3. Confusion matrix of unknown ICP detection. (a) Dataset 1. (b) Dataset 2.

TABLE 4. Performance of different methods in Dataset 1 and Dataset 2.

Dataset	Method	Accuracy	Precision	Score
Dataset 1	MGFAM-DL	0.9959	0.9958	0.9958
	1D-CNN	0.9515	0.9540	0.9527
	GRU	0.8599	0.8604	0.8601
	AM+1D-CNN+LSTM	0.9879	0.9842	0.9860
Dataset 2	MGFAM-DL	0.9843	0.9845	0.9843
	1D-CNN	0.8908	0.8575	0.8738
	GRU	0.7924	0.8132	0.8026
	AM+1D-CNN+LSTM	0.9725	0.9743	0.9734

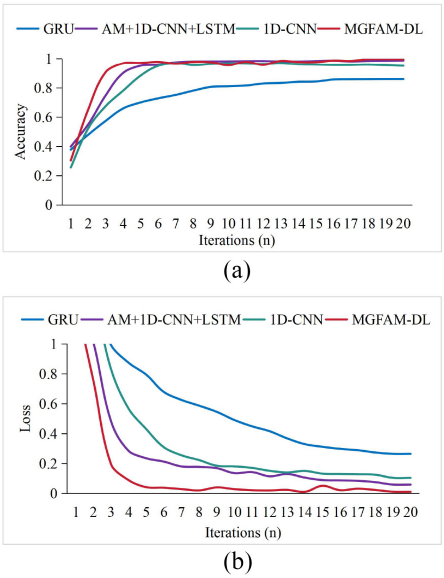


FIGURE 4. (a) Accuracy and (b) loss of different methods in Dataset 1.

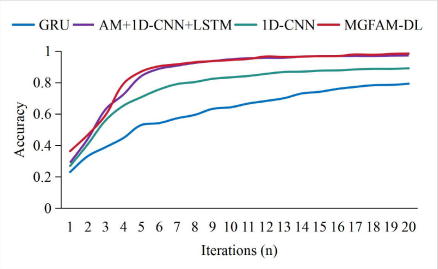
to other ICPs. Thus, it becomes challenging to correctly detect them.

According to Fig. 3, further calculate the precision and recall by (11) and (12). For Dataset 1, the precision of each category is 0.9917, 0.9918, 1, and 1, while the recall is 1, 0.9917, 1, and 0.9919. For Dataset 2, the precision of each category is 0.9665, 0.9817, 0.9900, and 1, while the recall is 0.9994, 0.9909, 0.9654, and 0.9816. It is evident that the proposed method achieves over 0.96 precision and recall for each category.

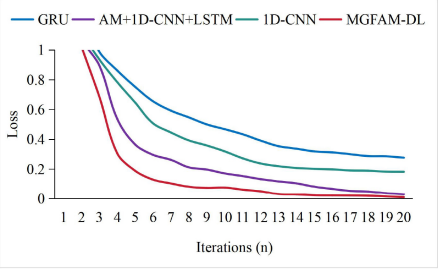
Furthermore, Table 4 comprehensively compares the accuracy, precision (mean), and score of different methods for Dataset 1 and Dataset 2. It can be observed that the proposed MGFAM-DL method exhibits the highest accuracy in both Dataset 1 and Dataset 2, where both accuracy and precision reach above 98.4%. Meanwhile, the performance of MGFAM-DL is better than that of simply 1D-CNN or GRU as MGFAM-DL combines the strengths of both 1D-CNN and GRU and can collectively optimize certain parameters during training, thereby enhancing the model's generalization ability.

Moreover, compared with AM-1D-CNN+LSTM, MGFAM-DL in Dataset 1 shows an improvement in accuracy by 0.8%, precision by 1.16%, and score by 0.98%, respectively. In contrast, the accuracy, precision, and score improvements in Dataset 2 are 1.19%, 1.03%, and 1.10%, respectively. This suggests that the performance of MGFAM-DL is a little better than that of AM-1D-CNN+LSTM. This is because both methods utilize a fusion approach of 1D-CNN and recurrent neural networks to extract features in ICPs. More importantly, MGFAM-DL is more better because it adopts the stacked 1D-CNN and GRU structure which allows for a more profound exploration of feature learning without the need for intricate structures. In contrast, AM-1D-CNN+LSTM introduces an attention mechanism, which provides better weight allocation but also increases the complexity of the network structure. This could potentially lead to increased difficulty in training the network structure, and in some cases, even result in overfitting, thereby impacting the accuracy.

Figs. 4 and 5 evaluate the accuracy and loss of different methods in different datasets. Obviously, MGFAM-DL achieves the rapid convergence with high accuracy, where convergence in Dataset 1 is only 3 iterations while that in Dataset 2 is 5 iterations. This demonstrates that the fusion of 1D-CNN and GRU possesses strengthened feature extraction capabilities, resulting in good training speed and fitting



(a)



(b)

FIGURE 5. (a) Accuracy and (b) loss of different methods in Dataset 2.

ability and enabling high accuracy to be achieved within a shorter time frame. GRU also displays a relatively smooth and gradual change in both accuracy and loss compared to 1D-CNN, without distinct abrupt changes. However, GRU and 1D-CNN converge at a slower rate compared to MGFAM-DL. This can be attributed to the lack of comprehensive feature extraction capabilities, resulting in poor performance. Additionally, both GRU and 1D-CNN have a low number of parameters and simpler network structures, necessitating a greater number of iterations for convergence during training. GRU exhibits a slower convergence rate compared to 1D-CNN due to its relatively weaker ability to extract ICP features. Moreover, AM-1D-CNN+LSTM also exhibits a rapid decrease in loss and improvement in accuracy after 5 and 7 iterations, showing a similar convergence speed as MGFAM-DL. However, the convergence speed is slightly worse than that of MGFAM-DL. This similarity can be attributed to the comprehensive feature extraction capabilities of both methods. The difference is due to the greater complexity of LSTM compared to GRU.

Fig. 6 demonstrates the running time of different methods on different datasets. In comparison to the single 1D-CNN and GRU, MGFAM-DL requires an increase in both computational workload and time consumption, as MGFAM-DL has to handle more layers and connections during the computational process. The MGFAM-DL increments with respect to 1D-CNN and GRU in Dataset 1 are 63.15% and 35.08%, respectively, while those in Dataset 2 are 73.54% and 40.46%, respectively. Despite there is an increased runtime of MGFAM-DL, it does not affect the operation of ICS. Furthermore, MGFAM-DL compared with AM-1D-CNN+LSTM, there is a reduction in runtime by 30.06% in Dataset 1 and 25.72% in Dataset 2. This is because

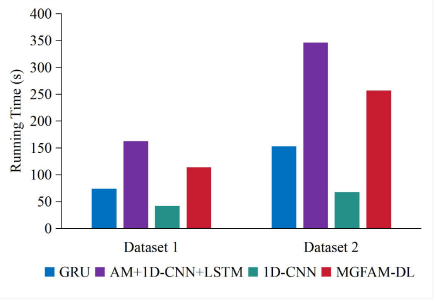


FIGURE 6. Running time of different methods.

GRU possesses simple internal mechanisms in MGFAM-DL, resulting in the reduction of computational workload and method parameters, thus decreasing the time consumption required for detection. Additionally, AM-1D-CNN+LSTM incorporates the attention mechanism, which on the one hand enhances the learning capability, but on the other hand impacts operational efficiency.

VI. CONCLUSION

In this article, the MGFAM-DL method was proposed to classify unknown ICPs for anomaly detection of ICSs. The method effectively extracted the features of ICP data by byte and half-byte multigranularity feature comparison, avoiding subjectivity in feature extraction. Furthermore, features were automatically marked, thereby resolving the issue of data labels. On this basis, the deep learning method combining 1D-CNN and GRU was employed to classify ICPs and achieve anomaly detection. The proposed method was compared with 1D-CNN, GRU, and AM-1D-CNN+LSTM by extensive experiments. The results demonstrated that the accuracy and precision of MGFAM-DL achieve above 98.4%, which is better than those of 1D-CNN, GRU, and AM-1D-CNN+LSTM.

Future works will consider explainable artificial intelligence methods to make anomaly detection more trustworthy and interpretative, such as the work for mobile traffic classification [24].

REFERENCES

[1] J.-C. Huang, G.-Q. Zeng, G.-G. Geng, J. Weng, K.-D. Lu, and Y. Zhang, "Differential evolution-based convolutional neural networks: An automatic architecture design method for intrusion detection in industrial control systems," *Comput. Security*, vol. 132, Sep. 2023, Art. no. 103310.

[2] C. Xu, X. Du, L. Li, X. Li, and H. Yu, "End-edge collaborative lightweight secure federated learning for anomaly detection of wireless industrial control systems," *IEEE Open J. Ind. Electron. Soc.*, vol. 5, pp. 132–142, 2024.

[3] J. Men, Z. Lv, X. Zhou, Z. Han, H. Xian, and Y.-N. Song, "Machine learning methods for industrial protocol security analysis: Issues, taxonomy, and directions," *IEEE Access*, vol. 8, pp. 83842–83857, 2020.

[4] W. Choi, S. Lee, K. Joo, H. J. Jo, and D. H. Lee, "An enhanced method for reverse engineering CAN data payload," *IEEE Trans. Veh. Technol.*, vol. 70, no. 4, pp. 3371–3381, Apr. 2021.

[5] C. Xu, P. Zeng, H. Yu, X. Jin, and C. Xia, "WIA-NR: Ultra-reliable low-latency communication for industrial wireless control networks over unlicensed bands," *IEEE Netw.*, vol. 35, no. 1, pp. 258–265, Jan./Feb. 2021.
[6] J. Narayan, S. K. Shukla, and T. C. Clancy, "A survey of automatic protocol reverse engineering tools," *ACM Comput. Surveys*, vol. 48, no. 3, pp. 1–26, 2015.
[7] Y. Huang, H. Shu, F. Kang, and Y. Guang, "Protocol reverse-engineering methods and tools: A survey," *Comput. Commun.*, vol. 182, pp. 238–254, Jan. 2022.
[8] S. Kleber, L. Maile, and F. Kargl, "Survey of protocol reverse engineering algorithms: Decomposition of tools for static traffic analysis," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 526–561, 1st Quart., 2019.
[9] K. L. X. Wang and B. Li, "IPART: An automatic protocol reverse engineering tool based on global voting expert for industrial protocols," *Int. J. Parallel, Emergent Distrib. Syst.*, vol. 35, no. 3, pp. 376–395, 2020.
[10] M. Marchetti and D. Stabili, "READ: Reverse engineering of automotive data frames," *IEEE Trans. Inf. Forensics Security*, vol. 14, pp. 1083–1097, 2019.
[11] B. Ning, X. Zong, K. He, and L. Lian, "PREIUD: An industrial control protocols reverse engineering tool based on unsupervised learning and deep neural network methods," *Symmetry*, vol. 15, no. 3, p. 706, 2023. [Online]. Available: https://www.mdpi.com/2073-8994/15/3/706
[12] J. Ling, Z. Zhu, Y. Luo, and H. Wang, "An intrusion detection method for industrial control systems based on bidirectional simple recurrent unit," *Comput. Electr. Eng.*, vol. 91, May 2021, Art. no. 107049.
[13] G. S. Sestito, A. C. Turcato, A. L. Dias, P. Ferrari, D. H. Spatti, and M. M. da Silva, "A general optimization-based approach to the detection of real-time Ethernet traffic events," *Comput. Ind.*, vol. 128, Jun. 2021, Art. no. 103413.
[14] C. Yu, Z. Zhang, and M. Gao, "An ICS traffic classification based on industrial control protocol keyword feature extraction algorithm," *Appl. Sci.*, vol. 12, no. 21, 2022, Art. no. 11193.
[15] J.-R. Jiang and Y.-T. Chen, "Industrial control system anomaly detection and classification based on network traffic," *IEEE Access*, vol. 10, pp. 41874–41888, 2022.
[16] K. Jin, L. Zhang, and D. Sun, "Research on pre-classification method of industrial control data based on adaptive BLOCK-DBSCAN," in *Proc. 42nd Chin. Control Conf. (CCC)*, 2023, pp. 8876–8881.
[17] H. Zhang, Y. Min, S. Liu, H. Tong, Y. Li, and Z. Lv, "Improve the security of industrial control system: A fine-grained classification method for DoS attacks on Modbus/TCP," *Mobile Netw. Appl.*, vol. 28, no. 2, pp. 839–852, Apr. 2023.
[18] R. Zhao and Z. Liu, "Analysis of private industrial control protocol format based on LSTM-FCN model," in *Proc. Int. Conf. Aviation Saf. Inf. Technol.*, New York, NY, USA, 2020, pp. 330–335.
[19] L. Zhai et al., "Identification of private ICS protocols based on raw traffic," *Symmetry*, vol. 13, no. 9, p. 1743, 2021.
[20] K. Lin, X. Xu, and H. Gao, "TSCRNN: A novel classification scheme of encrypted traffic based on flow spatiotemporal features for efficient management of IIoT," *Comput. Netw.*, vol. 190, May 2021, Art. no. 107974.
[21] X. Zhu, Z. Jiang, Q. Zhang, S. Zhao, and Z. Zhang, "An unknown protocol identification method for industrial Internet," *Wireless Commun. Mobile Comput.*, vol. 2022, Sep. 2022, Art. no. 3792205.
[22] S. Palekar and Y. Radhika, "IoT authentication model with optimized deep Q network for attack detection and mitigation," *Int. J. Intell. Robot. Appl.*, vol. 6, no. 2, pp. 350–364, Jun. 2022.
[23] "Network forensics and network security monitoring." Accessed: Sep. 15, 2021. [Online]. Available: https://www.netresec.com
[24] A. Nascita, A. Montieri, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescap, "XAI meets mobile traffic classification: Understanding and improving multimodal deep learning architectures," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 4, pp. 4225–4246, Dec. 2021.

XINYI DU received the B.S. degree in communications engineering from Liaoning Technical University, Huludao, China, in 2021, where she is currently pursuing the M.S. degree in communications and information systems.

Since 2021, she has been with the Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang, China. Her research interests include industrial control networks and 5G ultrareliable low-latency communications.

CHI XU (Senior Member, IEEE) received the Ph.D. degree in control theory and control engineering from the University of Chinese Academy of Sciences, Beijing, China, in 2017.

He is currently a Professor with the Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang, China. His research interests include industrial control networks, 5G URLLC, edge computing, and tactile Internet.

Prof. Xu is a Voting Member of the IEEE 1918.1 Working Group for Tactile Internet as well as a member of the IEEE 1932.1 Working Group for Licensed/Unlicensed Spectrum Interoperability in Wireless Mobile Networks. He also serves as a standardization delegate for 3GPP TSG RAN.

LIN LI received the M.S. degree in control theory and control engineering from Shenyang Ligong University, Shenyang, China, in 2016.

She is currently an Engineer with the Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang. Her research interests include industrial control system and robotics.

XINCHUN LI received the B.S. degree in electrical industrial automation from Liaoning Technical University, Huludao, China, in 1992.

He is a Senior Engineer with Liaoning Technical University. His research interests include embedded systems and wireless networks.