Delay and Energy-Efficient Asynchronous Federated Learning for Intrusion Detection in Heterogeneous Industrial Internet of Things

Shumei Liu, Yao Yu, *Member, IEEE,* Yue Zong, Phee Lep Yeoh, *Senior Member, IEEE,* Lei Guo, *Senior Member, IEEE,* Branka Vucetic, *Life Fellow, IEEE,* Trung Q. Duong, *Fellow, IEEE,* and Yonghui Li, *Fellow, IEEE*

***Abstract*—Federated learning (FL) is a promising solution to overcome data island and privacy issues in intrusion detection systems (IDS) for the Industrial Internet of Things (IIoT). However, the heterogeneity of various IIoT devices poses formidable challenges to FL-based intrusion detection, especially the training cost relating to delay and energy consumption. In this paper, we propose a delay and energy-efficient asynchronous FL (AFL) framework for intrusion detection (DEAFL-ID) in heterogeneous IIoT. Specifically, we address the shortcomings of low efficiency and high energy consumption in existing FL-based solutions involving all idle IIoT devices. To do so, we formulate an AFL-based optimal device selection problem which aims to select high-quality training devices in advance by exploring the device advantages in detection accuracy, delay reduction, and energy saving. Subsequently, a deep Q-network (DQN)-based learning algorithm is developed to quickly solve the above high-dimensional problem. In addition, to further improve the detection performance, we build a hybrid sampling assisted convolutional neural network (CNN)-based IDS model, which can eliminate the imbalance of IIoT data and enable the selected devices to fully extract data features. Through simulations, we demonstrate that DEAFL-ID achieves a significant improvement in training cost and detection performance compared with existing IDS schemes.**

***Index Terms*—Industrial Internet of Things (IIoT), intrusion detection, asynchronous federated learning (AFL), heterogeneous IIoT devices, delay and energy consumption.**

Manuscript received April 5, 2023; revised October 8, 2023; accepted December 13, 2023. This work was supported by the National Natural Science Foundation of China under Grant 62171113 and Grant 62301082. The work of T. Q. Duong was supported in part by the Canada Excellence Research Chair program. *(Corresponding author: Yao Yu.)*

S. Liu is with the School of Computer Science and Engineering, Northeastern University, Shenyang 110819, China, and also with the School of Information Engineering, Chang'an University, Xi'an 710064, China (E-mail: liusmneu@163.com).

Y. Yu is with both the School of Computer Science and Engineering, Northeastern University and Key Laboratory of Intelligent Computing in Medical Image, Ministry of Education, Northeastern University, Shenyang 110819, China (E-mail: yuyao@mail.neu.edu.cn).

Y. Zong is with the Power China Huadong Engineering Corporation Limited, Hangzhou, China (E-mail: zong_y2@hdec.com).

P. L. Yeoh is with the School of Science, Technology and Engineering, University of the Sunshine Coast, QLD 4502, Australia (e-mail: pyeoh@usc.edu.au).

L. Guo is with the School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China (E-mail: guolei@cqupt.edu.cn).

T. Q. Duong is with the Faculty of Engineering and Applied Science, Memorial University of Newfoundland, St. Johns, NL A1C 5S7, Canada, and is also with the School of Electronics, Electrical Engineering and Computer Science, Queens University Belfast, Belfast, U.K. (E-mail: tduong@mun.ca).

B. Vucetic and Y. Li are with the School of Electrical and Information Engineering, University of Sydney, Sydney NSW 2006, Australia (E-mail:{branka.vucetic; yonghui.li}@sydney.edu.au).

I. Introduction

THE Internet of Things (IoT) is an emerging paradigm with a wide span of interconnected smart devices and computing capabilities, and its specific applications have significantly changed the way we live [1]. Although IoT can increase productivity and efficiency through intelligent interconnection and remote management, it also increases the risk of cyber attacks due to the broadcast nature of wireless communications [2]. Especially in the industry 4.0 revolution, mission-critical industrial IoT (IIoT) systems support many anomaly-sensitive tasks such as industrial machinery operations and infrastructure control systems which require a high level of security to prevent attacks on critical operating devices and avoid system outages [3].

To alleviate the security vulnerability in IIoT devices, an effective IDS is essential to identify the possible attack through constantly analyzing IIoT data flows. Recently, learning-based IDSs have attracted significant attention due to their improved detection performance based on the learning and feature extraction from typical environmental data [4]. Traditionally, such IDS solutions can be categorized into two groups according to the different executors of data learning, namely local execution and server execution. In local execution [5], each device learns the intrusion detection model using its own private data. In reality, the IIoT environment is highly heterogeneous, which causes a wide variety of local data at different devices or at different locations. In this case, local execution cannot fully extract the data features in an industrial factory or department due to the one-sidedness of single-device data, and thus limiting the IDS performance. In contrast to local execution, server execution can overcome this by allowing a centralized server to collect local data from all devices within its coverage area and train a global intrusion detection model. In this case, the server manages the data of all IIoT devices, which is vulnerable to data privacy attacks and leads to a single point of failure. Consequently, IIoT data holders are often reluctant to share their training data with the central server to preserve data privacy. The resultant data island makes it challenging to achieve a satisfactory detection performance.

As such, there is an urgent need to develop a realistic learning-based IDS model to solve this formidable challenge. Very recently, federated learning (FL) has been considered as

a promising solution [6] to enable participating IIoT devices to build a global intrusion detection model without disclosing the underlying raw data [7]. In simple terms, FL allows devices to manage their own local data and send only certain parameters to the server for model aggregation. This provides a parallel scheme for devices to learn a global intrusion detection model collaboratively and protect their data privacy [8]. Typically, FL frameworks assume that all IIoT devices participate in detection model training and updating, but this assumption is not applicable to a heterogeneous industrial environment. In highly dynamic heterogeneous IIoT environments, the significantly different computation and communication resources of devices make it impossible for all IIoT devices to perform synchronous training and uploading of FL parameters [9]. As a consequence, a direct application of existing FL frameworks without any consideration of such heterogeneous properties will make the overall training process inefficient and unsuitable for rapid intrusion detection in IIoT.

Two limitations are crucial in this heterogeneous IIoT context. Firstly, at any time there will be a large number of idle devices with limited resources and requiring all of them to participate in the model training will lead to unnecessary waste of system resources. Secondly, the FL-based IDS will suffer from longer training times since some small IIoT devices have very limited computation capabilities and poor communication conditions. In such scenarios, the FL framework must wait for the upload parameter from the slowest device before aggregation, which compromises the training efficiency of the intrusion detection model. Therefore, traditional FL frameworks could result in severe performance bottlenecks for intrusion detection in heterogeneous IIoT environments. Given this background, the asynchronous federated learning (AFL) framework [10] is proposed to explore novel ideas for rapid model aggregation. Specifically, the conventional AFL approach is for the central server to perform global model aggregation as soon as it collects a local model. This is challenging for IIoT because the frequent model transfer aggregations result in massive communication resource consumption. For this limitation, one solution is to set a random deadline to terminate the upload step and ignore any updated parameters after the deadline. Of course, this simple approach will waste resources of delayed devices and could bias results to devices with fast training speed but low model accuracy. More recently in [11], the authors studied a client selection problem to minimize the training time of the FL process. But the authors in [11] focused only on training time and use the size of the local dataset as a rough proxy for client importance, which presents two challenges. On the one hand, due to the resource heterogeneity of IIoT devices, some devices have very limited computing resources and are more concerned with energy consumption than training time. Only considering the training time is not enough. On the other hand, due to the data heterogeneity in IIoT, the data among IIoT devices are not independent-and-identically-distributed, and thus the data size cannot directly represent the importance of IIoT devices. As such, there is a lack of a comprehensive AFL framework to handle the highly heterogeneous IIoT environment.

Motivated by the aforementioned discussion, in this paper we propose a delay and energy-efficient AFL framework for intrusion detection (DEAFL-ID) in heterogeneous IIoT. In our proposed DEAFL-ID scheme, we use prior knowledge (i.e., the previous device selection and model training results) to analyze the model training ability of the device or the quality of its local data. Our main goal is to select high-quality training IIoT devices in advance to overcome the shortcomings of low efficiency and high energy consumption. Here, the term "high-quality devices" refers to IIoT devices that have comprehensive advantages in terms of detection accuracy, training delay reduction, and training energy saving. Specifically, according to the heterogeneous computational capabilities, dynamic communication conditions and previous training results, we select the most suitable IIoT devices to participate in the current model updating. We believe the problems formulated and solved in this work are of great practical significance. The main technical contributions of this paper are as follows:

(1) We study the delay and energy efficiency advantages of AFL applied to intrusion detection in heterogeneous IIoT. Considering dynamic and heterogeneous IIoT, we aim to build a periodically updated intrusion detection system using an AFL framework. Due to concerns about both detection performance and model training cost, we focus on the whole intrusion detection process, including optimal device selection of AFL, data balance preprocessing, and convolutional neural network (CNN) model training.
(2) To improve the detection performance in the practical noisy IIoT environment, we design a hybrid sampling assisted CNN-based intrusion detection model, which can transform the practical unbalanced data set into a balanced one with no noise and clear classification boundaries. As such, each selected IIoT device can fully learn the various data features and quickly train a high-performance local intrusion detection model.
(3) To reduce training cost and concurrently ensure intrusion detection performance, we design a resource utilization efficiency (RUE) function to explore the gains of our proposed DEAFL-ID scheme in accuracy performance, delay reduction, and energy saving. Then, the formulated RUE maximization problem is equivalent to an optimal device selection problem subject to the selection state of currently idle IIoT devices. Subsequently, a DQN-based learning algorithm is developed to solve this high-dimensional problem.
(4) Through extensive numerical results, we demonstrate that our proposed DEAFL-ID scheme significantly outperforms the existing schemes in [23] and [24] in terms of detection performance and training cost of delay and energy consumption. Our solution is also proven to be more suitable for intrusion detection in heterogeneous IIoT.

The remainder of this paper is organized as follows. Section 2 introduces relevant state-of-the-art research. Section 3 introduces the system architecture and workflow of our proposed DEAFL-ID scheme. In Section 4, we present the hybrid

sampling assisted CNN-based intrusion detection model. In Section 5, we propose a delay and energy-efficient AFL framework for intrusion detection (DEAFL-ID), and formulate a RUE maximization device selection problem. We propose a DQN-based learning algorithm to solve our optimization problem in Section 6. In Section 7, we provide a detailed overview description of our DEAFL-ID scheme. We evaluate our proposed DEAFL-ID scheme by experimental simulation in Section 8. Finally, we conclude and discuss the paper in Section 9.

II. Related Works for Intrusion Detection

In recent years, we have witnessed an increasing research interest regarding intrusion detection in IoT [2]. Amongst the most popular intrusion detection methods, learning-based techniques have shown great potential for computer and IoT networks [12], [13]. In [12], the authors provided a comprehensive review of intrusion detection solutions deploying different aspects of learning techniques for IoT, and dealt with both traditional machine learning (ML) and reinforcement learning (RL) for intrusion detection. The authors in [13] introduced new regularizers into traditional ML to overcome the challenges posed by network data regarding high dimension, sparsity, and a lack of anomaly data. In [14], to improve the detection performance of traditional supervised learning approaches, the authors presented an active learning based method in IoT. However, traditional learning-based intrusion detection techniques may not be suitable in industrial IoT scenarios with an extremely large amount of data [15].

The authors in [16] indicated that simple ML-enabled solutions exhibit a lower performance and prediction accuracy for detecting intrusions in highly dynamic environments than that of deep learning-based solutions. To this end, they examined the use of optimization techniques to enhance the performance of a single-learner in intrusion detection. To accommodate the highly dynamic large-scale characteristics of the Internet of Vehicles (IoV), the authors in [17] considered an artificial intelligence (AI)-based intrusion detection architecture comprising deep learning engines for identification and classification of the vehicular traffic in the IoV networks. The deep learning-based intrusion detection model usually requires learning massive online environmental data, which is resource-consuming [18]. Therefore, to alleviate the burden of resource-constrained MEC servers, the authors in [19] proposed an intelligent intrusion detection algorithm for IoT using big data mining based on generative adversarial network (GAN) and CNN. However, these typically deep neural networks via conventional centralized training methods require aggregating all raw data to a central server, which will require large amounts of wireless bandwidth and may lead to privacy concerns [1].

To address the aforementioned shortcomings, researchers have proposed a new distributed model training framework known as FL [20]. In [21], the authors proposed an FL-based collaborative intrusion detection in vehicular edge computing, and blockchain was used to store the training models to ensure the security of the aggregation model. For large-scale and complex industrial systems, the authors in [3] proposed a deep learning scheme named DeepFed to detect cyber threats. In addition, [22] and [23] presented an FL-based intrusion detection algorithm to proactively recognize intrusions in IoT networks. The above schemes focus on the synchronous learning framework for intrusion detection, which incurs high communication energy costs while also leading to higher time waiting for slower nodes. More importantly, as we stated earlier, our work aims to design an intrusion detection system for heterogeneous IIoT where the IIoT devices always have a wide range of computation/communication resources and dynamic device conditions. These heterogeneous properties make it cumbersome for all IIoT devices to perform synchronous model training and updating. Also, the resulting intrusion detection model is inefficient and resource intensive. These challenges motivate us to develop a novel delay and energy-efficient AFL-based intrusion detection system for heterogeneous IIoT.

III. System Architecture and Workflow

As illustrated in Fig. 1, we consider an AFL-based intrusion detection framework for heterogeneous IIoT, where a MEC server manages many IIoT devices within a large-scale industrial area. Different from the traditional AFL framework [10], we only select some high-quality training devices in advance before model training. Compared with the traditional way of employing all idle IIoT devices, this can improve energy efficiency and reduce unnecessary waste of resources. Moreover, our solution considers the environmental heterogeneity of IIoT, and aims to build a periodically updated intrusion detection system using an AFL framework. Therefore, our solution is more suitable for resource-constrained devices in heterogeneous and dynamic IIoT. Our considered system framework comprises two types of entities, i.e., a MEC server and N idle IIoT devices at any time.

(1) *MEC Server:* The MEC server is a communication infrastructure (e.g., a wireless base station) with large amounts of computing/communication resources in the industrial IoT network area. Typically, a MEC server manages an industrial factory or several same-type factories within a given area. It is responsible for managing all the industrial devices in the covered area. Furthermore, in our AFL architecture, the MEC is responsible for selecting the IIoT devices for local model training in each iterative updating and building a comprehensive intrusion detection model by aggregating the local model parameters at the selected industrial agents. Multiple rounds of interactions between the MEC server and selected IIoT devices are demanded to obtain a final "perfect" intrusion detection model.

(2) *IIoT Devices:* IIoT devices are heterogeneous industrial devices with limited computing resources. They may access or leave the AFL architecture at any time due to the variability of the IIoT environment. Moreover, the devices managed by the MEC server generally have the same-type local data. Hence, all of these devices locally train a consistent intrusion detection model together. Each IIoT device trains a local intrusion detection model based on its own industrial data. Besides, it continuously

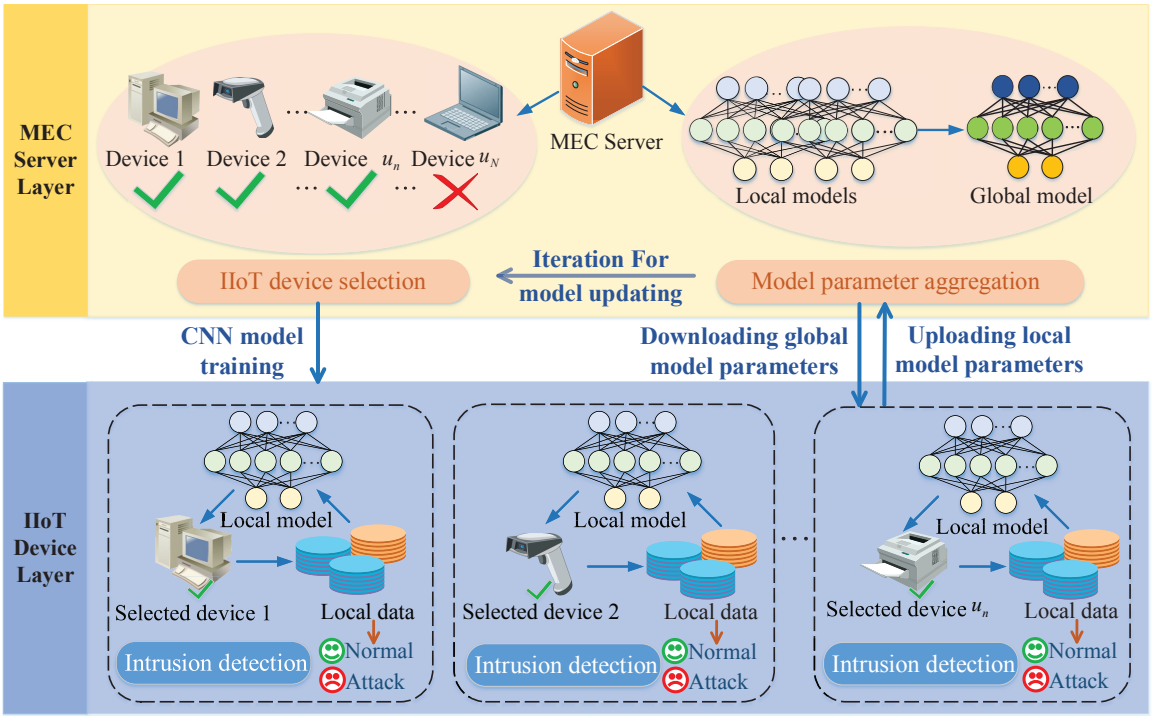


Fig. 1: AFL-based intrusion detection framework for heterogeneous IIoT

updates the local parameters of the intrusion detection model by recurrently interacting with the MEC server.

Next, we introduce the workflow of our AFL-based intrusion detection framework. For each iteration for periodic model updating, it involves the following steps:

(1) *System Initialization:* First, the MEC server determines the structure of the CNN model for intrusion detection. Specifically, it includes the number of convolutional layers, the number of maximum pooling layers, the size and number of filters in each layer, and the number of fully connected layers at the end of the model structure. Then, the MEC server uses public data to pre-train an initial intrusion detection model for local devices.

(2) *Selecting IIoT Devices by MEC Server:* Due to the rapid response requirements of the IIoT environment and the resource limitations of the IIoT devices, we focus on the delay and energy-efficient AFL for intrusion detection in heterogeneous IIoT networks. At the beginning of each iteration for model updating, we select some IIoT devices with better gains in training delay, energy consumption, and accuracy from the currently idle devices. Specifically, depending on the heterogeneous computational capacities, channel conditions, and model training accuracy of industrial devices, the MEC server adopts the DQN algorithm to select the most advantageous IIoT devices to participate in the current model training process.

(3) *Training Local Model by IIoT Devices:* After the selection of IIoT devices, each selected device locally trains a CNN-based intrusion detection model using their labeled private data. After that, each selected device uploads the currently trained model parameters and corresponding training accuracy to the MEC server. Furthermore, each selected device updates its own local model after receiving the global model parameters aggregated by the MEC server.

(4) *Aggregating Local Model by MEC Server:* During each model updating, the MEC server aggregates the local model parameters of the selected devices and interacts with the currently selected devices over multiple rounds to obtain a new global model parameter. Through multiple communication rounds, a convergent CNN-based intrusion detection model can finally be obtained.

(5) *Intrusion Detection at IIoT Devices:* Each IIoT device maintains a regularly updated local intrusion detection model. In addition, each local intrusion detection model always filters and monitors the local data of the device, as well as labels and classifies the data to obtain label data for later local model training and updating.

To sum up, our goal is to provide IIoT with an intrusion detection framework/model that can be fully learned, quickly monitored, and updated online, without limiting the data types of intrusion detection. The specific intrusions that can be identified depends on the inputted data type during model training and updating. In other words, for different IIoT, the model can be trained and updated online according to their respective network environment and data, and then the corresponding intrusion monitoring model can be customized.

IV. Hybrid Sampling Assisted CNN-Based Intrusion Detection Model

In this section, we build a hybrid sampling assisted CNN-based intrusion detection model for the local IIoT devices. CNN is a popular neural network that is typically used in image processing tasks. In recent years, due to its popularity

and prominent advantages, CNN is also extensively used in natural language processing, video analyzing, and intrusion detection. To better extract data features, we first propose a hybrid sampling method to preprocess the training data from two aspects of data balancing preprocessing and data standardization preprocessing. Then, we describe the CNN model structure and three main steps of intrusion detection.

A. Data Preprocessing Based on Hybrid Sampling

For intrusion detection, a reasonable data preprocessing method is essential. In reality, the generated data is an unbalanced data set, with most of it being normal data and only a small fraction containing the attack data that is critical to the system [24]. Due to the existence of boundary data and noise data, it may be difficult to detect the minority attack data during model training without data preprocessing. However, existing FL-based intrusion detection solutions have ignored the inherent unbalance and disarray of real data. To this end, we aim to propose a hybrid sampling method to transform the unbalanced data set into a balanced one with no noise and clear classification boundaries.

Specifically, for data balancing preprocessing, we present an improved Synthetic Minority Oversampling Technique (SMOTE) method based on K-means++ to alleviate the imbalance of training data, and employ Tomek Links [24] to eliminate noise. For data standardization preprocessing, we further perform symbolic feature numerization, data normalization, and data dimensionality reduction to obtain standard training data directly input to a CNN model. Next, we will detail these two aspects.

1) Data Balancing Preprocessing: Network traffic data comprises a large amount of normal traffic and a small amount of abnormal traffic, which is a typical imbalanced data classification problem. Moreover, real network data usually contain a lot of noise. In this case, the prediction accuracy of minority classes is often very low. Therefore, we aim to remove the noise while increasing the number of samples in the minority class.

(1) We divide the unbalanced dataset X into majority class X_M and minority class X_m.

(2) We use K-means++ algorithm [25] to obtain the data cluster centers of the minority class according to the following steps.

1) We randomly select a data sample from X_m as the first initial cluster center ω_1.

2) We calculate the Euclidean distance d_{i1} between each data x_i in X_m and the cluster center ω_1 as

$$d_{i1} = \|x_i - \omega_1\|_2. \tag{1}$$

3) We select the data point with the maximum d_{i1} in set X_m as the second initial cluster center ω_2. Then, the next initial cluster center is the one with the maximum value of $d_{i1} + d_{i2}$. Repeat the above process until c initial cluster centers are selected.

4) Based on the c cluster centers, we calculate the Euclidean distance d_{ij} between each remaining data x_i in set X_m and each cluster center ω_j using (1). Each data x_i is divided into the cluster C_j corresponding to the nearest cluster center ω_j (i.e., the minimum d_{ij}), and then c clusters are obtained.

5) For each cluster C_j, we recalculate its cluster center ω'_j as $\omega'_j = \arg\min J(\omega'_j)$, where

$$J(\omega'_j) = \sum_{x_i \in C_j} \|x_i - \omega'_j\|_2. \tag{2}$$

6) Repeat 4) and 5) until the cluster centers $\{\omega_1, ..., \omega_c\}$ no longer change. Then, we obtain the final c clusters $\{C_1, ..., C_c\}$. In this paper, the number of clusters is the number of sample categories in the minority class, and each cluster contains sample data of the corresponding category.

(3) We present an improved SMOTE method based on K-means++ to generate the minority class samples and alleviate the imbalance of training data.

To avoid the overfitting problem and improve the classification accuracy of the intrusion detection model, we fill more samples into the minority class using the over-sampling method. SMOTE is a commonly used over-sampling method [24]. The main idea is to form new minority class samples by interpolating between several samples that lie together. The new sample y_{new} can be obtained by

$$y_{\text{new}} = y_i + \text{gif} * (y_{i,j} - y_i), \tag{3}$$

where y_i is a data sample in minority class, $y_{i,j}$ is its jth nearest neighbor sample, and gif is a given random number. However, since the neighbor sample of the randomly selected sample y_i may be majority class, this traditional SMOTE method easily causes the decision boundaries of the minority class to spread into the majority class space, and then directly affects the accuracy of the trained intrusion detection model. To solve the above problem, we present an improved SMOTE method based on K-means++. Specifically, based on the cluster centers obtained by step (2), we use sample y_i and cluster center c_i in cluster C_i to fill the sample data of the minority class, which can greatly reduce the probability of the new sample spreading to the majority class. Then, new sample y_{new} can be obtained by

$$y_{\text{new}} = y_i + \text{gif} * (y_i - c_i), \tag{4}$$

as such, we expand the number of minority class samples, and the imbalanced training dataset can be transformed into a balanced one. The reason we use K-means++ instead of K-means is that the K-means++ algorithm presents less randomness, and the obtained multiple cluster centers are less correlated. In this way, for different minority samples, the samples we fill are less relevant. As such, we make the boundaries of different types of samples clearer, which is convenient for model training and data feature extraction.

(4) We use Tomek Links to eliminate noise and boundary samples.

In reality, network traffic data contain a lot of noise. Besides, some borderline data may be synthesized during the data balancing processing process. The above data is considered as "bad" since it may cause the model to identify on the wrong side of the decision border. Therefore, to improve the classification accuracy of the trained model, Tomek links are used as an under-sampling method to remove noise and borderline data.

Based on the above data balancing preprocessing process, a balanced data set with low noise and clear classification boundaries is obtained.

2) Data Standardization Preprocessing: The process of data standardization preprocessing includes symbolic feature numerization, data normalization, and data dimensionality reduction.

(1) Symbolic Feature Numerization

Almost all labeled training data is symbolic data, such as the NSL-KDD dataset [24], which is commonly used in the intrusion detection field. Since the input of the CNN model is a numerical matrix, we convert the non-numeric symbolic features in the training dataset into numeric features by using the one-hot encoder.

(2) Data Normalization

Data normalization can eliminate the differences between different dimensional data and is therefore widely used in computing. In this paper, we apply the min-max normalization, and uniformly linearly map the range of values of each feature in the interval [0, 1]. The transformation function is given by

$$x_{i2} = \frac{x_{i1} - x_{\min}}{x_{\max} - x_{\min}}, \tag{5}$$

where x_{i1} represents the initial data i of a feature, $x_{\max}$ represents the maximum data of the feature, $x_{\min}$ represents the minimum data of the feature, and x_{i2} represents the data after normalizing x_{i1}.

(3) Data Dimensionality Reduction

In general, the initial data format in the training dataset, such as NSL-KDD, is a $1 * n$ dimension vector. If the initial data is directly fed to the CNN model as inputs, it will jeopardize the real-time requirement due to the increased complexity of computation. Therefore, considering that the CNN model has great classification performance for image data, we use a dimension reduction algorithm and convert the $1 * n$ vector into $m * m$ image data format, where $n = m * m$.

Through the above hybrid sampling, we alleviate the imbalance of training data, eliminate noise and boundary samples, and standardize data features. As such, we can help the model fully learn the data features and improve the model training accuracy.

B. CNN-Based Intrusion Detection Model

After data preprocessing, the processed data is supplied as an input of the local model for training. In this paper, we adopt CNN for the local intrusion detection model thanks to its excellent performance in automatic feature extraction. Specifically, we consider the classic CNN model LeNet-5 [26] as the intrusion detection model, which mainly contains two convolutional layers, two pooling layers, and two fully connected layers. In terms of functionality, the convolutional layers and pooling layers together handle the feature extraction of the training samples while the fully connected layer performs the final classification. Next, we detail the process of extracting data features using CNN.

First, the preprocessed data matrix is fed into the first convolutional layer. The convolutional layer is the core of the CNN structure. It can extract different features of the data through different convolution kernels. In the convolutional operation, the same convolutional kernel follows the principle of "parameter sharing", which can greatly reduce the number of parameters and increase the computational efficiency. The convolutional function is written as

$$q_i = f(q_{i-1} \otimes v_i + b_i), \tag{6}$$

where q_i is the feature map of layer i, $f(\cdot)$ is the activation function, $\otimes$ is the convolutional function, v_i is the convolutional kernel of layer i, and b_i is the bias of layer i.

Next, the pooling layer works after the convolution layer. It usually samples the feature map following different sampling rules. The pooling layer has two purposes: (i) reducing network burden by reducing the feature dimension, and (ii) avoiding over-fitting by filtering out redundant features. The pooling processes can be expressed as

$$q_i = pool(q_{i-1}), \tag{7}$$

where $pool(\cdot)$ is the sampling function.

After two convolutional and pooling layers, the fully connected layer converts the 2D data feature to 1D and then passes it through the classifier for classification. Through the above CNN processes, we can output the features of the training traffic data.

For intrusion detection, the pre-processed data is divided into the training set and test set, of which the training set accounts for a large proportion (usually 80%). Then, the CNN-based intrusion detection model includes the following three steps.

(1) *Training:* Each IIoT device trains the CNN model using the local training data. Specifically, the goal of training is to improve the detection accuracy of the CNN model through constant adjustment of model parameters.
(2) *Testing:* After the training stage, each IIoT device checks the training accuracy of the CNN model using test data. Specifically, if the model satisfies the training accuracy requirement or reaches the predetermined training rounds, the training step would cease; otherwise, the model would repeat the training step.
(3) *Evaluating:* Finally, the model performance needs to be evaluated. In this paper, the considered evaluation metrics include *Accuracy*, *Precision*, *Recall*, and *F1-Score*.

V. Delay and Energy-Efficient AFL Framework for Intrusion Detection

In this section, we explore the efficiency advantages of AFL applied to intrusion detection in heterogeneous IIoT. We first abstract the training process of the CNN model into a local computing model. Then, we describe the local computing and communication model at the heterogeneous IIoT devices. Next, we design the device selecting model of AFL, and formulate the corresponding optimization framework.

A. Local Computing and Communication Model

In this subsection, we explain the local computing and communication model at each IIoT device. For local computing, we formulate the training delay and energy consumption of devices with different computing capabilities during the local CNN training process. For the communication between the IIoT devices and the MEC server, we formulate the transmission delay and energy consumption during the AFL parameter uploading process.

1) Computing Model: The time required for IIoT device u_n to complete CNN training during each iteration can be expressed as

$$T_n^{\text{comp}} = \frac{D_n g_n}{f_n}, \tag{8}$$

where D_n represents the number of data samples in the local training data set of device u_n, g_n is the number of CPU cycles required to train one data sample on device u_n, and f_n is the CPU frequency (in CPU cycles per second) of device u_n. Due to the heterogeneous computing capacities of IIoT devices, the CPU frequency f_n of different devices varies greatly. Furthermore, the energy consumption [20] for training D_n data samples at device u_n is written as

$$E_n^{\text{comp}} = \varsigma_n g_n D_n f_n^2, \tag{9}$$

where $\varsigma_n > 0$ is the effective capacitance coefficient for the chip processor architecture of device u_n.

2) Communication Model: After the local training is completed, the selected IIoT device u_n sends the trained local model parameter (i.e., weight and offset) to the MEC server for further parameter aggregation. The achievable transmission rate r_n of device u_n can be expressed using Shannon's theory as

$$r_n = B_n \log_2\left(1 + \frac{h_n p_n}{n_0}\right), \tag{10}$$

where p_n is the transmission power of device u_n, and n_0 is the background noise. Here, h_n and B_n respectively are the channel gain and bandwidth of u_n, and dynamic h_n and B_n reflect the time-varying communication condition of different devices. Then, the communication time for sending model parameters from device u_n to the MEC server is given by

$$T_n^{\text{comm}} = \frac{\delta_n}{r_n}, \tag{11}$$

where δ_n is the local model parameter size of device u_n. Based on (11), the energy consumption of device u_n is $E_n^{\text{comm}} = p_n T_n^{\text{comm}}$.

B. Device Selection Model of AFL

The existing FL-based intrusion detection solutions default to a synchronization process involving all idle devices, which is time- and energy-consuming. Moreover, in heterogeneous IIoT, the significantly different computing resources of devices and time-varying communication conditions make it non trivial for all idle devices to perform synchronous training and parameters uploading. Therefore, we propose a delay and energy-efficient AFL framework for intrusion detection (DEAFL-ID), and mainly focus on the optimal device selection of the AFL process. Specifically, we define the time gain and energy gain indexes and accuracy gain index to measure the comprehensive advantages of IIoT devices in terms of delay reduction, energy saving, and detection accuracy. To this end, we introduce a binary security variable $\lambda_n^k = \{0, 1\}$ to represent the selection decision of IIoT device u_n at the kth iteration. $\lambda_n^k = 1$ means that device u_n is selected for local model training during the kth iteration, and otherwise $\lambda_n^k = 0$. Next, we will describe the device selection process in more detail.

1) Time Gain and Energy Gain Indexes: Regarding our considered AFL for intrusion detection, the total time consumption T_k^{AFL} and the total energy consumption E_k^{AFL} during the k-th iterative training process can be expressed as

$$T_k^{\text{AFL}} = \max_{u_n \in \mathcal{U}_k} \lambda_n^k (T_n^{\text{comp}} + T_n^{\text{comm}}) \tag{12}$$

$$E_k^{\text{AFL}} = \sum_{u_n \in \mathcal{U}_k} \lambda_n^k (E_n^{\text{comp}} + E_n^{\text{comm}}), \tag{13}$$

where $\mathcal{U}_k = \{u_1, ..., u_n, ..., u_N\}$ is the set of N currently idle IIoT devices.

In addition, we introduce the total time T_k^{FL} and energy consumption E_k^{FL} of classical FL during the k-th training and parameter uploading process, respectively, as follows:

$$T_k^{\text{FL}} = \max_{u_n \in \mathcal{U}_k} (T_n^{\text{comp}} + T_n^{\text{comm}}) \tag{14}$$

$$E_k^{\text{FL}} = \sum_{u_n \in \mathcal{U}_k} (E_n^{\text{comp}} + E_n^{\text{comm}}). \tag{15}$$

As such, the time gain and energy gain indexes of DEAFL-ID during the k-th device selection process can be defined as $\mathcal{T}_k = \frac{T_k^{\text{AFL}}}{T_k^{\text{FL}}}$ and $\mathcal{E}_k = \frac{E_k^{\text{AFL}}}{E_k^{\text{FL}}}$, respectively.

2) Accuracy Gain Index: For the $k-1$th device selection (the $k-1$th device selection corresponds to the $k-1$th periodic model updating), we assume that $\mathcal{Z}_{k-1}$ is the set of selected devices. Given the number of communication rounds O, the local model parameter vector of device u_n is $\mathbf{w}_n^{\text{loc},k-1} = (w_{n,1}^{\text{loc},k-1}, ..., w_{n,o}^{\text{loc},k-1}, ..., w_{n,O}^{\text{loc},k-1})$ and the global model parameter vector is $\mathbf{w}^{\text{glo},k-1} = (w_1^{\text{glo},k-1}, ..., w_o^{\text{glo},k-1}, ..., w_O^{\text{glo},k-1})$, where $o \in \{1, 2, ..., O\}$ represents the oth communication round. Then, we measure the Euclidean distance between $\mathbf{w}_n^{\text{loc},k-1}$ and $\mathbf{w}^{\text{glo},k-1}$, which is defined by $\beta_n^{k-1} = ||\mathbf{w}_n^{\text{loc},k-1} - \mathbf{w}^{\text{glo},k-1}||_2$. A smaller β_n^{k-1} means that the local parameters of device $u_n \in \mathcal{Z}_{k-1}$ are closer to the global parameters. Thus, device u_n has better local training data or greater model training advantage during the $k-1$th model updating process. When performing

the kth device selection, we consider the previous τ times (i.e., from $k-\tau$ to $k-1$) device selection results, and pay attention to the O rounds local training situation of the selected devices in each model updating process. For each device $u_n \in \mathcal{Z}_{k-i}, \forall i \in \{1,2,...,\tau\}$, we set

$$\widetilde{\beta}_n^{k-i} = \alpha_{k-i} \cdot \beta_n^{k-i} = \alpha_{k-i} \cdot ||\mathbf{w}_n^{\mathrm{loc},k-i} - \mathbf{w}^{\mathrm{glo},k-i}||_2, \quad (16)$$

where $\alpha_{k-1} > \alpha_{k-2} > ... > \alpha_{k-\tau}$ and $\alpha_{k-1} + \alpha_{k-2} + ... + \alpha_{k-\tau} = 1$. Here, we set decreasing weighted value for the Euclidean distances from $k-1$ to $k-\tau$. This is because the idle devices participating in the kth device selection process and their local data are closest to those of $k-1$, and conversely the most different from those of $k-\tau$.

As such, for each device $u_n \in \mathcal{U}_k$, we define its accuracy gain index as

$$\mathcal{V}_k = \frac{\sum_{u_n \in \mathcal{U}_k} \lambda_n^k \cdot \upsilon_n^k}{\sum_{u_n \in \mathcal{U}_k} \upsilon_n^k} \quad (17)$$

$$\upsilon_n^k = \begin{cases} \frac{\widetilde{\beta}_n^{k-1} - \widetilde{\beta}_{\min}}{\widetilde{\beta}_{\max} - \widetilde{\beta}_{\min}}, \text{ if } u_n \in \mathcal{Z}_{k-1} \\ \frac{\widetilde{\beta}_n^{k-2} - \widetilde{\beta}_{\min}}{\widetilde{\beta}_{\max} - \widetilde{\beta}_{\min}}, \text{ if } u_n \notin \mathcal{Z}_{k-1} \text{ and } u_n \in \mathcal{Z}_{k-2} \\ \quad \vdots \qquad\qquad \vdots \\ \frac{\widetilde{\beta}_n^{k-\tau} - \widetilde{\beta}_{\min}}{\widetilde{\beta}_{\max} - \widetilde{\beta}_{\min}}, \text{ if } u_n \notin \mathcal{Z}_{k-1} \cup \mathcal{Z}_{k-2} \cup ... \cup \mathcal{Z}_{k-\tau+1} \\ \qquad\qquad\qquad \text{and } u_n \in \mathcal{Z}_{k-\tau} \\ 0.5, \qquad \text{else } u_n \notin \mathcal{Z}_{k-1} \cup \mathcal{Z}_{k-2} \cup ... \cup \mathcal{Z}_{k-\tau} \end{cases} \quad (18)$$

where $\widetilde{\beta}_{\min} = \min\{\widetilde{\beta}_j^{k-i}, \forall u_j \in \mathcal{Z}_{k-i}, \forall i \in \{1,2,...,\tau\}\}$ and $\widetilde{\beta}_{\max} = \max\{\widetilde{\beta}_j^{k-i}, \forall u_j \in \mathcal{Z}_{k-i}, \forall i \in \{1,2,...,\tau\}\}$. Moreover, 0.5 represents a random guess for the unknown devices in $\mathcal{U}_k$. We find that $\mathcal{V}_k$ is a normalized variable that ranges from 0 to 1. Therefore, it is fair to set $\mathcal{V}_k$ to 0.5 for devices whose performance is unknown because they have not been selected. 0.5 is recognized by scholars as a random guess for unknown performance. In addition, our device selection framework is a long-term selection process. For each round of device selection, we consider the results of the previous rounds of selections, and many devices are previously selected (i.e. devices with known training performance). Since $\mathcal{V}_k$ is a normalized indicator that considers all previously selected devices, for those devices that perform well in model training accuracy, their $\mathcal{V}_k$ value must be greater than 0.5, and the best performing devices have a value of 1.

Based on the above analysis, to highlight the advantages of our DEAFL-ID scheme, we define a Resource Utilization Efficiency (RUE) function for the kth device selection process of AFL, which represents the relative improvement in detection accuracy, training time and energy consumption according to

$$\mathrm{RUE}_k = \frac{\mathcal{V}_k}{\mathcal{T}_k + \mathcal{E}_k}, \quad (19)$$

For k-th device selection, RUE_k is equal to the accuracy gain index of the selected devices divided by the sum of the time gain index and the energy gain index of the selected devices. For this case, the device selection scheme will converge to the strategy of selecting devices with both accuracy efficient and time/energy efficient. Obviously, the RUE of the classical FL-based intrusion detection solution without device selection is a fixed value of 0.5. Subsequently, to reduce the training time and energy consumption of the training process while ensuring the accuracy of the intrusion detection model, we aim to maximize RUE_k of the AFL process in each iteration k. Given the computing capacities f_n and communication conditions B_n and h_n of all idle IIoT devices, our device selection problem is formulated as the following maximization problem with respect to λ_n^k

$$\max_{\lambda_n^k} \ \mathrm{RUE}_k \quad (20a)$$

$$\text{s.t.} \ \ \lambda_n^k \in \{0,1\}, \ \forall u_n \in \mathcal{U}_k. \quad (20b)$$

As such, instead of using all idle devices, some highly qualified devices are chosen to train and upload the models, which fully combines the advantages of AFL and the heterogeneous characteristics of IIoT, thereby improving the resource utilization efficiency of the system. It helps to reduce training costs incurred by the delay and energy consumption, while ensuring detection accuracy. However, the problem in (20) is a nonlinear integer programming and nonconvex problem due to the nonlinear and high complexity of our objective function (20a).

VI. Problem Transformation and Solution by Deep Reinforcement Learning

Since optimization problem (20) is NP-hard, traditional optimization methods usually require high computational complexity to obtain suboptimal solutions. To this end, we use a deep reinforcement learning (DRL) algorithm - DQN to tackle this problem. DQN focuses on maximal long-term rewards through learning from the network state and making an optimal decision. It can find a good or even optimal policy directly from historical observations without any requirement to know the system dynamics.

A. Problem Transformation

We attempt to transform the above RUE maximization problem into the Q value maximization problem of DQN. First, we define the state space, action space and reward function in the DQN process as follows:

(1) *State Space* $\mathcal{S}$

At each step t of the DQN network training, The state space includes the following:

1) The CPU frequencies of devices $\mathbf{f}(t) = \{f_1(t), ..., f_n(t), ...f_N(t)\}$;

2) The channel gains of devices $\mathbf{h}(t) = \{h_1(t), ...h_n(t), ...h_N(t)\}$;

3) The number of data samples in devices $\mathbf{D}(t) = \{D_1(t), ..., D_n(t), ...D_N(t)\}$;

4) The computation bandwidths of devices $\mathbf{B}(t) = \{B_1(t), ..., B_n(t), ..., B_N(t)\}$;

5) For each device u_n, the local model parameters of previous τ-round completed device selections (if is selected) $\mathbf{W}_n^{\mathrm{loc}} = \{\mathbf{w}_n^{\mathrm{loc},k-\tau}, ..., \mathbf{w}_n^{\mathrm{loc},k-i}, ..., \mathbf{w}_n^{\mathrm{loc},k-1}\}$. For all idle devices, the local model parameters of previous τ-round completed device selections (if is selected) is $\mathbf{W}^{\mathrm{loc}} = \{\mathbf{W}_1^{\mathrm{loc}}, ..., \mathbf{W}_n^{\mathrm{loc}}, ..., \mathbf{W}_N^{\mathrm{loc}}\}$;

6) The global model parameters of previous τ-round completed device selections $\mathbf{W}^{\text{glo}} = \{\mathbf{w}_n^{\text{glo},k-\tau}, ..., \mathbf{w}_n^{\text{glo},k-i}, ..., \mathbf{w}_n^{\text{glo},k-1}\}$;

7) The selection state of last step $t-1$ for devices $\boldsymbol{\lambda}(t-1) = \{\lambda_{1,t-1}, ..., \lambda_{n,t-1}, ..., \lambda_{N,t-1}\}$.

As such, the system state $s(t) \in \mathcal{S}$ can be defined as $s(t) = \{\mathbf{f}(t), \mathbf{h}(t), \mathbf{D}(t), \mathbf{B}(t), \mathbf{W}^{\text{loc}}, \mathbf{W}^{\text{glo}}, \boldsymbol{\lambda}(t-1)\}$.

(2) *Action Space* $\mathcal{A}$

The action at step t is the device selection decision, which can be regarded as a binary problem. The action $a(t) \in \mathcal{A}$ is defined by a vector as $a(t) = \{\lambda_{1,t}, ..., \lambda_{n,t}, ..., \lambda_{N,t}\}$, where $\lambda_{n,t} = 1$ means that device u_n is selected as the participating devices of AFL. Otherwise $\lambda_{n,t} = 0$.

(3) *Reward Function* $\mathcal{R}(s, a)$

Based on the above states and actions, we can obtain the time gain $\mathcal{T}_k$, energy gain $\mathcal{E}_k$, and accuracy gain $\mathcal{V}_k$ of current devices. Then, we obtain $\text{RUE}(t)$ at each training step t of device selection. The system evaluates the effect of an action by using the reward function $\mathcal{R}(s(t), a(t))$. It can obtain an immediate reward when taking action $a(t)$ at state $s(t)$. Then, the system reward of device selection at step t can be represented as $\mathcal{R}(s(t), a(t)) = \text{RUE}(t)$.

(4) *Policy* π

Let $\pi : \mathcal{S} \to \mathcal{A}$ denote the policy function, which maps any state $s(t) \in \mathcal{S}$ to action $a(t) \in \mathcal{A}$. In step t, the action to be taken can be obtained by the policy $a(t) = \pi(s(t))$.

(5) *Next State* $s(t+1)$

After taking action $a(t)$ at state $s(t)$, the system states transit from $s(t)$ to $s(t+1)$, where $s(t+1) \Leftarrow s(t) + \pi(s(t))$. The new updated state includes $\mathbf{f}_n(t)$, $\mathbf{h}_n(t)$, $\mathbf{B}_n(t)$ and $\boldsymbol{\lambda}(t-1)$.

Consequently, by using the Q-function, solving problem (18) is equivalent to finding the optimal policy π^* to maximize the total cumulative reward, which can be expressed as

$$Q^*(s, a) = \max_{\pi^*} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s(t), \pi^*(s(t)))\right], \quad (21)$$

where $\mathbb{E}[\cdot]$ is the mathematical expectation, Q^* is the optimal Q-function, and $\gamma \in [0, 1]$ is the discount factor. If the optimal Q-function value is determined, the optimal policy (i.e., the optimal device selection strategy) is given by

$$\pi^*(s) = \arg\max_{a \in \mathcal{A}} Q^*(s, a), \forall s \in \mathcal{S}. \quad (22)$$

B. Problem Solution

We drop the symbol of time slot t and use s' to represent the next state. Then, based on (19) and (20), the Q-learning algorithm is iterated by updating the Q-function value as

$$Q(s, a) = \mathcal{R}(s, a) + \gamma \max_{a' \in \mathcal{A}} Q(s', a'). \quad (23)$$

The Q-learning algorithm is inefficient for solving complex problems with high dimensions and high dynamics [1]. To this end, the further evolved DQN algorithm establishes two neural networks with the same structure and different parameters, namely Q evaluate (i.e., main) network and Q target network, to solve the instability of Q-learning. As such, DQN can extract complex features among various states and actions. The Q main network is trained to enable $Q(s, a, \theta) \approx Q_\pi(s, a)$, where θ is the network weight of the main network. In each learning iteration, we train the neural network to the minimize the updated function $L(\theta)$ which is defined by

$$\mathcal{L}(\theta) = \mathbb{E}\left[(\mathcal{R}(s, a) + \gamma \max_{a' \in \mathcal{A}} \bar{Q}(s', a'; \theta^-) - Q(s, a; \theta))^2\right], \quad (24)$$

where θ^- is the network weight of the target networks, and $\bar{Q}(\cdot)$ is the Q-function of the target network. Differentiating the loss function with respect to the weight θ, the gradient is given by

$$\nabla_\theta \mathcal{L}(\theta) = \mathbb{E}\Big[2(\mathcal{R}(s, a) + \gamma \max_{a' \in \mathcal{A}} \bar{Q}(s', a'; \theta^-) - Q(s, a; \theta)) \times \nabla_\theta Q(s, a, \theta)\Big]. \quad (25)$$

Based on the gradient descent, θ is updated as

$$\theta \leftarrow \theta + \sigma \nabla_\theta \mathcal{L}(\theta), \quad (26)$$

where σ is a step size coefficient that controls the updating step size in each iteration.

Our proposed DQN-based device selection strategy is illustrated in Algorithm 1. Based on the trained deep neural networks in Algorithm 1, we input the states of all currently idle IIoT devices. Then, the output would be the device selection results of our proposed DEAFL-ID scheme.

Complexity analysis: For the DQN-based device selection algorithm, the training process runs offline and is performed at the MEC server which has sufficient computation resource. Hence, we focus our analysis on the complexity for online decision making. The total complexity of Algorithm 1 is incurred by calculating the output based on the input and finding the action with the maximal Q value. Then, the complexity of Algorithm 1 is theoretically analyzed as follows. In Algorithm 1, the input state comprises the number of currently idle devices N, the CPU frequencies of the devices, the channel gains of the devices, the computation bandwidths of the devices, and the initial selection state of the devices. Hence, the number of input neurons for the DQN is $4N$. There are N neurons in the output layer. The number of hidden layers is denoted as L, and each hidden layer has H neurons. As such, the number of multiplication operations for the DQN is $4N \times H + (L-1) \times H \times H + H \times N = \mathcal{O}(LH^2 + 5NH - H^2)$ and can be simplified into $\mathcal{O}(LH^2)$. Moreover, the complexity of applying the activation function is $\mathcal{O}(L \times H)$. Therefore, the complexity of computing the DQN output is $\mathcal{O}(LH^2 + LH)$ and can be simplified into $\mathcal{O}(LH^2)$. In addition, the operation for selecting the action with the maximum Q value is with complexity $\mathcal{O}(N)$. Based on the above analysis, the total complexity is $\mathcal{O}(LH^2)$.

Convergence analysis: In the following, we discuss the convergence property of our AFL framework. As highlighted above, the asynchrony of our solution is reflected in device selection. Then, after each device selection, the selected devices participate in the federated learning process, and train a convergent model together. Next, we analyze the convergence of the federated learning process. We define the upper bound of

Algorithm 1 DQN-Based Device Selection Algorithm

Initialize experience replay memory $\mathcal{D}$
Initialize Q-function with random weights θ
Initialize $\bar{Q}$-function with weights $\theta^- = \theta$
Initialize $\varepsilon = 0.1$, $\gamma = 0.9$ and $\lambda_0 = [1, ..., 1]$
for each episode j **do**
  Initialize the parameters in environment setup;
  for each step t **do**
    Input initial state and obtain all actions;
    Choose a random probability e;
    Perform a ε-greedy policy as $a(t) = \begin{cases} \text{a randomly selected action, if } e \le \varepsilon \\ \arg\max_{a \in \mathcal{A}} Q(s(t), a(t); \theta), \quad \text{otherwise} \end{cases}$;
    Execute $a(t)$, and obtain $\mathcal{R}(s(t), a(t))$ and $s(t+1)$;
    Store transition $\{s(t), a(t), \mathcal{R}(s(t), a(t)), s(t+1)\}$ into $\mathcal{D}$;
    Randomly sample transitions from $\mathcal{D}$;
    Calculate the output of DQN as $Y_j = \begin{cases} \mathcal{R}_j, \text{ if episode terminates at } j+1 \\ \mathcal{R}_j + \gamma \max_{a' \in \mathcal{A}} \bar{Q}(s', a'; \theta^-), \text{otherwise} \end{cases}$
    Calculate loss according to $\mathcal{L}_j(\theta) = \mathbb{E}[(Y_j - Q(s, a; \theta))^2]$;
    Perform the gradient descent on $\mathcal{L}_j(\theta)$ with respect to θ using (25);
    Update θ using (26);
    Every ι steps, update θ^- with θ;
  end for
end for
return The parameters of the trained deep neural networks.

the divergence between the federated loss function $F(\mathbf{w})$ and the optimal global loss function $F(\mathbf{w}*)$ as $|F(\mathbf{w}) - F(\mathbf{w}*)|$, where $\mathbf{w}*$ is the optimal global model parameters. As such, the convergence of the federated learning process can be proved if it satisfies $|F(\mathbf{w}) - F(\mathbf{w}*)| \le \iota$, where $\iota > 0$ is a small positive constant. According to [11] and [27], when $F(\mathbf{w})$ is a η-convex and σ-smooth function, the upper bound of $F(\mathbf{w}) - F(\mathbf{w}*)$ can be expressed $F(\mathbf{w}) - F(\mathbf{w}*) \le \iota(F(\mathbf{w}(0)) - F(\mathbf{w}*))$, where $\mathbf{w}(0)$ is initial model parameter. Therefore, for given appropriate communication rounds O, the federated learning process will finally converge to the global optimality. The specific proof analysis is similar to the FL convergence proof in [27].

VII. Solution Overview

In this section, we provide a detailed overview description of our proposed DEAFL-ID scheme. Specifically, we present a visual flow diagram to illustrate the main steps in DEAFL-ID, as show in Fig. 2. Considering the environmental heterogeneity of IIoT, we aim to build a periodically updated intrusion detection system using an AFL framework. Fig. 2 shows the solution workflow of kth system updating process. We next detail the process.

(1) *List the Current Set of Idle Devices*: Due to the high priority of industrial tasks, for each IDS updating the devices in busy states do not participate in the training process. As such, we need to list the devices in idle states before model updating and training (i.e., considering whether the devices have time to participate in the model training).

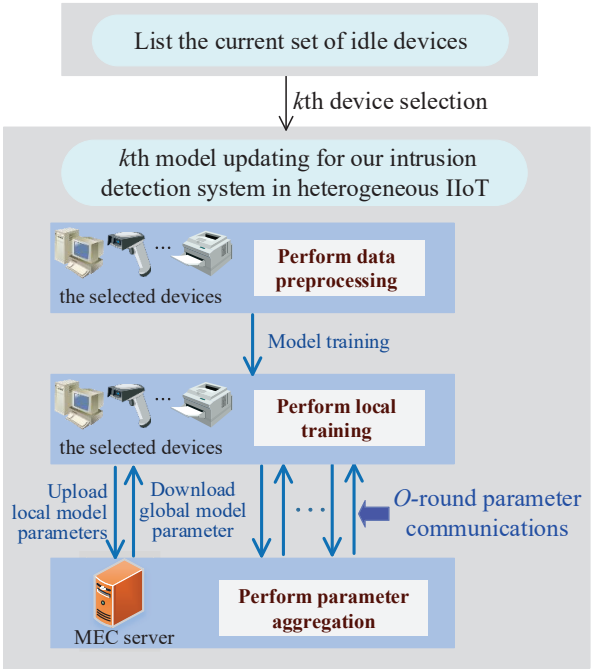


Fig. 2: Solution workflow of the proposed DEAFL-ID scheme

(2) *Model Updating for Intrusion Detection Systems:* For the current idle IIoT devices, we use prior knowledge (i.e., the previous device selection and model training results) to analyze the model training ability of the device or the quality of its local data. Based on this, our goal is to select the "high-quality" devices that have comprehensive advantages in terms of detection accuracy, training delay reduction, and training energy saving. The details of device selection are given in Sections 5 and 6.

1) *Perform data preprocessing:* Since the data in reality is the unbalanced data set with unclear boundaries and noise data, it may be difficult to detect the minority attack data during model training without data preprocessing. To this end, we propose a hybrid sampling method to transform the unbalanced data set into a balanced one with no noise and clear classification boundaries. The details of technical process is shown in Section 4.
2) *Perform local training:* Each selected device locally trains a CNN-based intrusion detection model using their labeled private data. After that, each selected device uploads the currently trained model parameters to the MEC server.
3) *Perform parameter aggregation:* The MEC server aggregates the local model parameters of the selected devices and then send the aggregated global model parameters to them. The MEC server interacts with the currently selected devices over O rounds

to obtain a new intrusion detection model.

VIII. Performance Evaluation

In this section, we evaluate the performance of the proposed DEAFL-ID scheme via extensive simulations. All the simulations are conducted in Python on a desktop computer with an Intel Core i7-7700 3.60GHz CPU and 16GB RAM.

We first describe the simulation setup. Then we validate the performance of our DEAFL-ID scheme from three aspects: convergence analysis, training cost analysis, and intrusion detection performance analysis. As mentioned earlier, due to concerns about both model training cost and detection performance, we focus on the whole intrusion detection process including optimal device selection of AFL and data balance preprocessing assisted CNN model training. Therefore, to verify our DEAFL-ID, the two other schemes used for comparisons are configured as follows:

(1) FL-based intrusion detection scheme without data preprocessing (FL-ID-NoDP) [23]: This scheme is derived from [23]. For a fair comparison, this scheme also adopts the CNN model as the local intrusion detection model. Then, it utilizes FL without device selection to iteratively train and update the intrusion detection model, and finally obtains an optimal global model. As such, this framework must wait for the upload parameter from the slowest device before model aggregation. Moreover, since there is no device selection, the framework cannot discard model parameters for devices that have disadvantages in terms of training time and accuracy.

(2) AFL-based intrusion detection scheme with data preprocessing method in [24] (AFL-ID-DP): This scheme is derived by combining the data preprocessing method in [24] and our AFL framework. Specifically, we use the data preprocessing method in [24] to replace our proposed hybrid sampling-based data preprocessing method in DEAFL-ID. To deal with data imbalances, this scheme adopts the traditional SMOTE method shown as equation (3) and interpolated the new samples between the minority class data and its neighbor sample.

A. Simulation Setup

Before analyzing the simulation results, we detail the stimulation parameter settings and training dataset in our simulation experiment.

1) Simulation Parameters: The simulation parameters settings are summarized in Table 1 unless otherwise stated. Specifically, we set different D_n, f_n, p_n, B_n and h_n within the given ranges as shown in Table 1. As such, we can simulate the heterogeneous IIoT devices with different data resources (i.e., D_n), computational capacities (i.e., f_n and p_n) and channel conditions (i.e., B_n and h_n). Furthermore, we use a truncated Gaussian distribution to simulate the varying computational and communication state, that is $f_n(t) \sim \mathcal{N}(f_n, 0.1)$, $B_n(t) \sim \mathcal{N}(B_n, 0.1)$, and $h_n(t) \sim \mathcal{N}(h_n, 0.1)$.

TABLE I: Simulation parameter settings

Parameters	Values
Background noise at AP: n_0	-50dbm
CPU cycles required to process 1bit data: g_n	20 cycles/bit
Effective capacitance coefficient: ς_n	2×10^{-28}
CPU frequency of device u_n: f_n	$[1 \sim 3] \times 10^9$ cycles/second
The size of data samples: D_n	$[4 \sim 6]$ Mb
The size of local model parameter: δ_n	0.01 Mb
Channel bandwidth: B_n	$[2 \sim 10]$ MHz
Channel power gain: h_n	$[0.5 \sim 1] \times 10^{-7}$
Transmission power: p_n	$[0.2 \sim 0.5]$ W

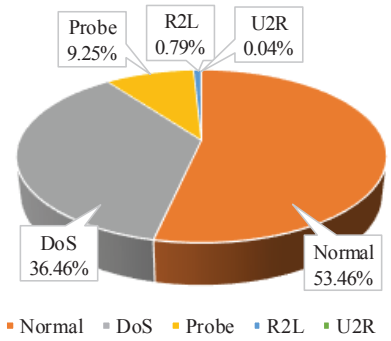
2) Training Dataset: We use the NSL-KDD Train Set [28] to implement and evaluate the proposed DEAFL-ID scheme. It is divided into 80% and 20% subsets for training and testing, respectively. The training set was divided equally into 30 portions, with each participating user owning one portion. NSL-KDD contains 41 features labeled as normal or attack type. They can be categorized as one normal class and four attack classes. The four attack classes are further grouped as Denial-of-Service (DoS), Probe, Root to Local (R2L), and Unauthorized to Root (U2R). NSL-KDD is a typical unbalanced dataset because the number of R2L and U2R attack samples is very small. More importantly, R2L and U2R attacks are hidden in the data payload of packets. From the outside, packets containing these two attacks look no different from normal packets. As such, the minority attacks R2L and U2R are difficult to identify. We highlight that the purpose of IDSs is to detect more covert and deadly attacks. To this end, our proposed hybrid sampling method can transform the imbalanced training dataset into a balanced one, and use it as our training set. Consequently, before and after hybrid sampling, the number of various types' records in the training set is shown in Fig. 3. Compared to the original dataset in Fig. 3 (a), the one after our proposed hybrid sampling in Fig. 3 (b) achieves the inter-class balance between majority and minority classes and intra-class balance within the minority class. As such, the selected IIoT devices can fully learn the various data features in the dataset.

3) CNN Model Structure and Evaluation Metrics: According to the feature dimensions of the NSL-KDD data set, this paper sets the structure of the CNN for intrusion detection as shown in TABLE 2. We consider the four key metrics of *Accuracy*, *False positive rate*, *Precision*, *Recall* and *F1-Score* to evaluate the performance of the intrusion detection model.

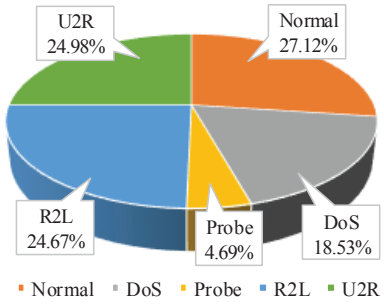
B. Simulation Results and Analysis

We first evaluate the reward convergence and loss of the proposed DQN-based device selection algorithm. In addition, to analyze the effect of device selection, we analyze the training cost including time and energy consumption of AFL-based intrusion detection. Finally, we test the detection performance of our DEAFL-ID scheme.

1) Convergence Analysis: In this case, we assume that the number of currently idle IIoT devices is 10. Their computation



(a) Original NSL-KDD



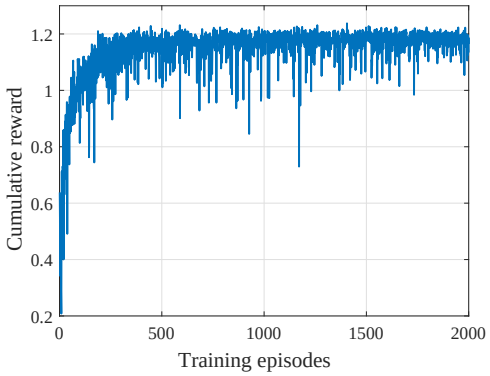
(b) After hybrid sampling

Fig. 3: The proportion of various attack types in the NSL-KDD dataset

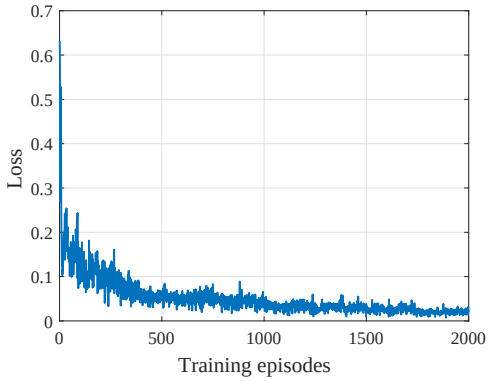
TABLE II: CNN model structure and parameter settings

Layer	Type	Kernel/ Pool size	Strides	Activation function	Output
L1	Input	-	-	-	1*11*11
L2	Convolutional layer 1	3*3, 32 filters	1	Relu	32*11*11
L3	Pooling layer 1	2*2	2	Relu	32*5*5
L4	Convolutional layer 2	3*3, 64 filters	1	Relu	64*5*5
L5	Pooling layer 2	2*2	2	Relu	64*2*2
L6	Fully connected layer 1	-	-	Relu+ Dropout	128
L7	Fully connected layer 2	-	-	Softmax	5

capacities and communication conditions are randomly generated based on the given ranges in Table 1. Then, we use our proposed Algorithm 1 to obtain the optimal device selection strategy. The convergence performance of the proposed DQN-based device selection algorithm is shown in Fig. 4. We can observe that the reward and loss quickly converge to the near-optimal values within the 500 training episodes. Moreover, with the increase of training episodes, the reward obtained by Algorithm 1 gradually increases and the loss gradually decreases, indicating that the training process will not fall into overfitting. Therefore, we can conclude that our proposed Algorithm 1 has good convergence performance.



(a) Reward analysis



(b) Loss analysis

Fig. 4: Convergence performance of the proposed DQN-based device selection algorithm

2) Training Cost Analysis: Next, we evaluate the training cost performance of the model training process by comparing our proposed DEAFL-ID scheme with the benchmark scheme FL-ID-NoDP in [23] (In this case, the results of AFL-ID-NoDP is the same as our DEAFL-ID scheme since it is derived from DEAFL-ID). The computation capacities and communication conditions of idle devices are randomly generated based on the given ranges in Table 1 for each experimental trial, and we present the average results for a large number of trials.

Fig. 5 shows the number of selected IIoT devices versus the number of the currently idle IIoT devices in the two schemes, and the values on the data lines represent the detection accuracy of the resulting global model. We find that the numbers of selected and idle devices are positively correlated. This is because as the number of idle devices increases, the server will select more devices to participate in model training to improve the accuracy of intrusion detection model. Moreover, it is obvious that our proposed DEAFL-ID scheme outperforms the FL-ID-NoDP scheme due to the smaller number of selected IIoT devices and the similar global model accuracy. Employing fewer number of devices frees them to perform other industrial tasks and improve their resource utilization. The reason for the improvement is that DEAFL-ID selects devices with higher detection accuracy, and lower training delay and energy consumption, which is

considerably different from FL-ID-NoDP that requires all idle devices to participate in model training.

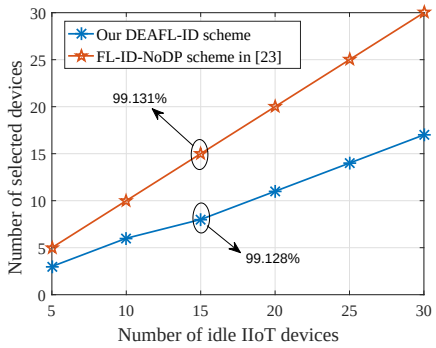
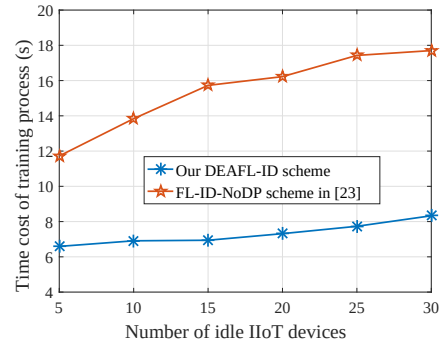


Fig. 5: Number of selected devices versus number of idle IIoT devices

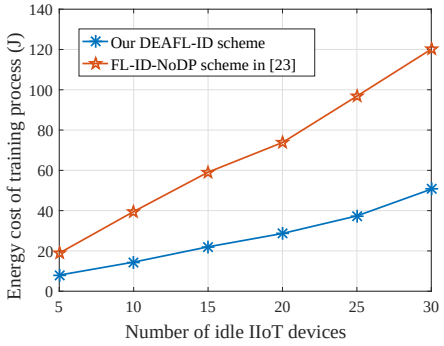
Figs. 6 (a) and (b) show the time and energy cost of the training process changing with the number of currently idle IIoT devices in the two schemes, respectively. In Fig. 6 (a), we observe that our proposed DEAFL-ID scheme outperforms the FL-ID-NoDP scheme, which shows the high efficiency of our solution. There are two reasons for this improvement. First, we select only some devices instead of all idle IIoT devices to participate in model training, and recall that the slowest device dominates the training time of each round. Therefore, if more devices are selected, there is a higher probability of including slow devices. Second, we give priority to selecting those IIoT devices with better gains in the delay reduction. In Fig. 6 (b), we see that our proposed DEAFL-ID scheme outperforms the FL-ID-NoDP scheme. This is because we select some IIoT devices with better gains in energy saving instead of all idle IIoT devices to participate in model training. This improvement is explained from the fact that more participating devices will lead to higher energy consumption in the training process. More importantly, according to (19), we can obtain that the average RUE of our DEAFL-ID scheme is about 0.93 while the FL-ID-NoDP scheme in [23] is only 0.5. Therefore, we conclude that our intrusion detection solution is more suitable for heterogeneous IIoT with low-delay requirements and limited-resource devices because of the low training cost and high RUE.

3) Intrusion Detection Performance Analysis: Finally, we evaluate the detection performances of the IDS by comparing our proposed DEAFL-ID scheme with the two benchmark schemes of FL-ID-NoDP and AFL-ID-DP. In this case, we set 10 currently idle IIoT devices. The computation capacities and communication conditions of these devices are randomly generated based on the given ranges in Table 1 for each experimental trial, and we present the average results for a large number of trials.

Table 3 shows the different intrusion detection performances for precision, recall, and F1-score with different data types in the three schemes, and Fig. 7 shows the average performances of the schemes. From Table 3 and Fig. 7, we can observe that the different detection performances of our DEAFL-ID scheme are close to 99% and always outperforms the other schemes in



(a) Time cost



(b) Energy cost

Fig. 6: Training cost of training process changing with the number of the currently idle devices

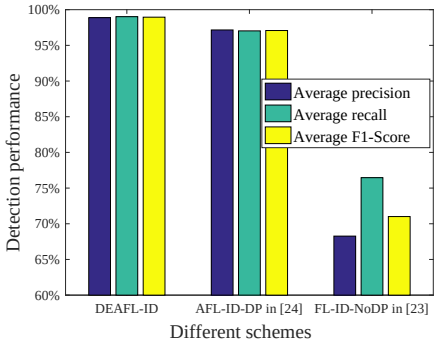


Fig. 7: Average performances of different schemes

all the considered performance metrics. This improvement is mainly due to the hybrid sampling method for data preprocessing in our DEAFL-ID scheme. The FL-ID-NoDP scheme does not have a data preprocessing step, which makes it difficult for them to find the minority class data and easy to classify errors due to the presence of environmental noise. The AFL-ID-DP scheme in [24] proposed a data preprocessing method, that fills more samples into the minority class using the SMOTE method. Hence, AFL-ID-DP has better classification performance than FL-ID-NoDP. However, AFL-ID-DP adopts the traditional SMOTE method given in equation (3) and interpolates the new samples between the minority class data and its neighbor sample. Since the neighbor sample may be

TABLE III: Detection performances regarding precision, recall and F1-Score to NSL-KDD

	Our DEAFL-ID scheme			AFL-ID-DP scheme in [24]			FL-ID-NoDP scheme in [23]		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Normal (%)	98.88	98.57	98.73	97.95	95.75	96.84	98.07	97.80	97.93
DoS (%)	99.44	99.44	99.44	99.66	97.79	98.72	99.51	98.13	98.82
Probe (%)	96.88	97.72	97.30	96.87	96.65	96.75	97.27	96.60	96.93
R2L (%)	99.22	99.36	99.29	94.55	97.27	95.89	46.48	89.75	61.20
U2R (%)	99.97	99.98	99.99	96.79	97.70	97.24	0	0	0
Average (%)	98.88	99.02	98.95	97.16	97.03	97.08	68.27	76.46	71.00

from the majority class, this approach may cause the decision boundaries of the minority class to spread into the majority class space and then decreases the classification accuracy of the trained intrusion detection model. By contrast, in DEAFL-ID we present an improved SMOTE method based on K-means++. Specifically, we identify the data cluster centers of the minority class. Then, we use sample y_i and cluster center c_i in cluster C_i to fill the sample data of the minority class as equation (4), which can greatly reduce the probability of the new sample spreading to the majority class and achieve the best classification performance among the three schemes.

Table 4 shows the detection performances regarding accuracy and false positive rates in the three schemes. Benefiting from our proposed hybrid sampling-based data preprocessing method, DEAFL-ID outperforms the other two schemes. By contrast, for the AFL-ID-DP scheme in [24], the sample spreading results in unclear data sample boundaries, and thus it achieves the worst accuracy and false positive rates among the three schemes. Moreover, for the FL-ID-NoDP scheme in [23], although its classification performances in Table 3 are poor given the fact that it struggles to identify the minority sample, it has little influence on the overall accuracy performance due to the very small proportion (0.83% in NSL-KDD) of minority class samples.

TABLE IV: Detection performances regarding accuracy and false positive rates

Different schemes	Our DEAFL-ID scheme	AFL-ID-DP scheme in [24]	FL-ID-NoDP scheme in [23]
Accuracy rate	99.25%	97.07%	97.75%
False alarm rate	1.42%	4.25%	2.20%

Fig. 8 shows the accuracy performance changing with communication rounds between the MEC server and the selected IIoT devices in the three schemes. We can find that the accuracy in different schemes increases with the number of communication rounds. This is because the MEC server performs model aggregation at each round and sends a newer and better global model to the selected devices. Moreover, DEAFL-ID quickly achieves the near-optimal values within 9 communication rounds, which verifies the convergence of our AFL framework. Furthermore, our proposed DEAFL-ID scheme always outperforms the other schemes, and the reason is the same as Table 4.

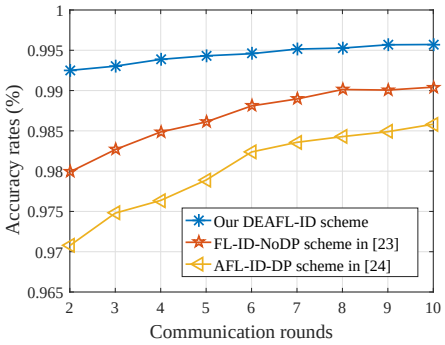


Fig. 8: Accuracy changing with communication rounds between the server and selected devices

Fig. 9 shows the accuracy performance versus the number of currently idle IIoT devices in the three schemes. It is obvious that our DEAFL-ID scheme performs better than the other schemes, due to the same reasons discussed for Table 4. Moreover, we find that the accuracy in DEAFL-ID and FL-ID-NoDP in [23] increases at first and then slightly decreases with the increasing number of idle devices. The early increase in performance is because the more devices will bring more training data and improve the training accuracy of the model. Then, when the number of idle devices reaches a certain number, a continued increase in the number of devices adversely affects the accuracy performance in the three schemes. This degradation is because when the training data set reaches a certain size, the model obtained by training has reached close to the optimal accuracy. After that, more devices bring various updated parameters, making it difficult for the server to aggregate them to a better global model. This is one of the reasons the AFL-ID-DP scheme in [24] decreases with the increasing number of idle devices. The most fundamental reason for AFL-ID-DP's decline is the sample spreading caused by the data preprocessing method in [24]. This further leads to unstable model parameters, and more participating devices result in worse model aggregation performance. We note that our DEAFL-ID scheme always outperforms the other schemes regardless of the number of idle IIoT devices.

To sum up, benefiting from our delay and energy-efficient AFL framework and hybrid sampling method for data preprocessing, we can significantly reduce the model training cost

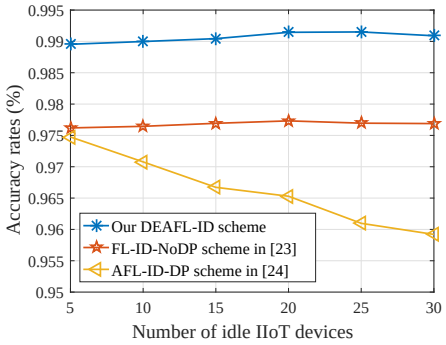


Fig. 9: Accuracy changing with the number of idle IIoT devices

as well as improve the model detection accuracy and classification performance in heterogeneous IIoT environments.

IX. Conclusion and Future Works

For realistic heterogeneous IIoT networks, an effective IDS plays an essential role in ensuring network security. Amongst the popular solutions, learning-based techniques have proven their great potential in IoT networks. However, the existing learning-based methods depend on collecting the local data from all participating devices, which leads to privacy concerns. While FL-based IDS have been considered to address the privacy problem, it is still challenging to implement in the IIoT due to its heterogeneous nature and limited device resources. To this end, we proposed a DEAFL-ID scheme to provide an efficient and high-performance intrusion detection for heterogeneous IIoT. Specifically, we design a CNN-based hybrid sampling assisted intrusion detection model to fully extract the features of large-scale unbalanced IDS data. Moreover, we developed a device selection optimization framework by exploring the gains of our solution in detection accuracy, delay reduction, and energy saving compared with the traditional FL-based intrusion detection. Then, we develop a DQN-based learning algorithm to find the optimal device selection policy. Through extensive simulation results, we verified that our proposed DEAFL-ID scheme can significantly outperform benchmark schemes.

In future studies, we will continue to improve this work. For example, we will consider and use some specific IIoT data such as the dataset proposed in [29] to test and analyze the proposed intrusion detection framework in this paper. Moreover, we will consider some non-i.i.d. (independent and identically distributed) data to conduct the simulation analysis.

References

[1] B. Li, J. Liu and B. Ji, "Low-Overhead Wireless Uplink Scheduling for Large-Scale Internet-of-Things," *IEEE Trans. Mobile Comput.*, vol. 20, no. 2, pp. 577-587, Feb. 2021.

[2] S. Nandy, M. Adhikari, M. A. Khan, V. G. Menon and S. Verma, "An Intrusion Detection Mechanism for Secured IoMT framework based on Swarm-Neural Network," *IEEE Journal of Biomedical and Health Informatics*, Early Access, DOI: 10.1109/JBHI.2021.3101686, Aug. 2021.

[3] B. Li, Y. Wu, J. Song, R. Lu, T. Li and L. Zhao, "DeepFed: Federated Deep Learning for Intrusion Detection in Industrial Cyber-Physical Systems," *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5615-5624, Aug. 2021.

[4] A. Jamalipour and S. Murali, "A Taxonomy of Machine-Learning-Based Intrusion Detection Systems for the Internet of Things: A Survey," *IEEE Internet Things J.*, vol. 9, no. 12, pp. 9444-9466, Jun. 2022.

[5] B. Naik, M. S. Obaidat, J. Nayak, D. Pelusi, P. Vijayakumar and S. H. Islam, "Intelligent Secure Ecosystem Based on Netaheuristic and Functional Link Neural Network for Edge of Things," *IEEE Trans Ind. Informat.*, vol. 16, no. 3, pp. 1947-1956, Mar. 2020.

[6] E. Bakopoulou, B. Tillman and A. Markopoulou, "FedPacket: A Federated Learning Approach to Mobile Packet Classification," *IEEE Trans. Mobile Comput.*, vol. 21, no. 10, pp. 3609-3628, Oct. 2022.

[7] S. A. Rahman, H. Tout, C. Talhi and A. Mourad, "Internet of Things Intrusion Detection: Centralized, On-Device, or Federated Learning?" *IEEE Network*, vol. 34, no. 6, pp. 310-317, Sep. 2020.

[8] J. Xu and H. Wang, "Client selection and bandwidth allocation in wireless federated learning networks: A long-term perspective," *IEEE Trans. Wireless Commun.*, vol. 20, no. 2, pp. 1188-1200, Oct. 2020.

[9] Y. Lu, X. Huang, K. Zhang, S. Maharjan and Y. Zhang, "Blockchain Empowered Asynchronous Federated Learning for Secure Data Sharing in Internet of Vehicles," *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 4298-4311, Apr. 2020.

[10] C. Xu, Y. Qu, Y. Xiang and L. Gao, "Asynchronous Federated Learning on Heterogeneous Devices: A Survey," arXiv preprint arXiv:2109.04269, 2021.

[11] W. Xia, T. Q. S. Quek, K. Guo, W. Wen, H. H. Yang and H. Zhu, "Multi-Armed Bandit-Based Client Scheduling for Federated Learning," *IEEE Wireless Commun.*, vol. 19, no. 11, pp. 7108-7123, Nov. 2020.

[12] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac and P. Faruki, "Network Intrusion Detection for IoT Security Based on Learning Techniques," *IEEE Commun. Surveys Tuts.*, vo. 21, no. 3, pp. 2671-2701, Aug. 2019.

[13] V. L. Cao, M. Nicolau, and J. McDermott, "Learning neural representations for network anomaly detection," *IEEE Trans. Cybern.*, vol. 49, no. 8, pp. 3074C3087, Aug. 2019.

[14] K. Yang, J. Ren, Y. Zhu and W. Zhang, "Active learning for wireless IoT intrusion detection," *IEEE Wireless Commun.*, vol. 25, no. 6, pp. 19-25, Dec. 2018.

[15] R. Heartfield, G. Loukas, A. Bezemskij and E. Panaousis, "Self-Configurable Cyber-Physical Intrusion Detection for Smart Homes Using Reinforcement Learning," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 1720-1735, Dec. 2020.

[16] G. Abdelmoumin, D. B. Rawat and A. Rahman, "On the Performance of Machine Learning Models for Anomaly-Based Intelligent Intrusion Detection Systems for the Internet of Things," *IEEE Internet Things J.*, Early Access, DOI 10.1109/JIOT.2021.3103829, Aug. 2021.

[17] T. Alladi, V. Kohli, V. Chamola, F. R. Yu and M. Guizani, "Artificial Intelligence (AI)-Empowered Intrusion Detection Architecture for The Internet of Vehicles," *IEEE Wireless Commun.*, vol. 28, no. 3, pp. 144-149, Feb. 2021.

[18] L. Nie, Z. Ning, X. Wang, X. Hu, J. Cheng and Y. Li, "Data-Driven Intrusion Detection for Intelligent Internet of Vehicles: A Deep Convolutional Neural Network-Based Method," *IEEE Transactions on Network Science and Engineering*, vol. 7, pp. 4, pp. 2219-2230, Apr. 2020.

[19] Y. Wu, L. Nie, S. Wang, Z. Ning and S. Li, "Intelligent Intrusion Detection for Internet of Things Security: A Deep Convolutional Generative Adversarial Network-enabled Approach," *IEEE Internet Things J.*, Early Access, DOI: 10.1109/JIOT.2021.3112159, Sep. 2021.

[20] N. H. Tran, W. Bao, A. Zomaya, M. N. H. Nguyen and C. S. Hong, "Federated Learning over Wireless Networks: Optimization Model Design and Analysis," in *Proc. IEEE INFOCOM*, Location: Paris, France, May 2019.

[21] H. Liu, S. Zhang, P. Zhang, X. Zhou, X. Shao, G. Pu and Y. Zhang, "Blockchain and Federated Learning for Collaborative Intrusion Detection in Vehicular Edge Computing," *IEEE Trans. Veh. Technol.*, vol. 70, no. 6, pp. 6073-6084, Jun. 2021.

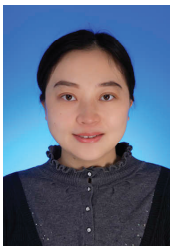
[22] Z. Chen, N. Lv, P. Liu, Y. Fang, K. Chen and W. Pan, "Intrusion Detection for Wireless Edge Networks Based on Federated Learning," *IEEE Access*, vol. 8, pp. 217463-217472, Dec. 2020.

[23] V. Mothukuri, P. Khare, R. M. Parizi, S. Pouriyeh, A. Dehghantanha G. and Srivastava, "Federated Learning-based Anomaly Detection for IoT Security Attacks," *IEEE Internet Things J.*, vol. 9, no. 4, pp. 2545-2554, Feb. 2022.

[24] K. Jiang, W. Wang, A. Wang and H. Wu, "Network Intrusion Detection Combined Hybrid Sampling With Deep Hierarchical Network," *IEEE Access*, vol. 8, pp. 32464-32476, Feb. 2020.

[25] S. Ben-David, D. Pal and H. U. Simon, "Stability of k-means clustering," in *Proc. International conference on computational learning theory*, Springer, Berlin, Heidelberg, pp. 20-34, Jun. 2007.
[26] A. Sayal, S. T. Nibhanupudi, S. Fathima and J. P. Kulkarni, "A 12.08-TOPS/W All-Digital Time-Domain CNN Engine Using Bi-Directional Memory Delay Lines for Energy Efficient Edge Computing," *IEEE J. Solid-State Circuits*, vol. 55, no. 1, pp. 60-75, Jan. 2020.
[27] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive Federated Learning in Resource Constrained Edge Computing Systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 446C452, pp. 1205-1221, Jun. 2019.
[28] L. Dhanabal and S. P. Shanthara, "A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms," *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 4, no. 6, pp. 446C452, Jun. 2015.
[29] M. Al-Hawawreh, E. Sitnikova and N. Aboutorab, "X-IIoTID: A Connectivity-Agnostic and Device-Agnostic Intrusion Data Set for Industrial Internet of Things," *IEEE Internet of Things Journal*, vol. 9, no. 5, pp. 3962-3977, Mar. 2022.

Shumei Liu received the B.S. degree in electronic and information engineering from Shanxi University, Taiyuan, China, in 2016 and the M.S. degree in electronics and communication engineering from Northeastern University, Shenyang, China, in 2018. She received the Ph.D. degree in communication and information systems from Northeastern University in 2022. She is currently a lecturer at the School of Information Engineering, Chang'an University, Xi'an, China. Her research interests include the Unmanned Aerial Vehicles (UAV), mobile edge computing, and wireless resource management. She has received the Best Paper Award in National Postdoctoral Academic Forum in China (2018).



Yao Yu (Member, IEEE) received B.S. degree in communication engineering from the Northeastern University, Shenyang, China in 2005, and the Ph.D. degree in communication and information system from the Northeastern University, Shenyang, China in 2010. From 2010 to 2011, she was a Postdoctoral Fellow with Department of Computing at Hong Kong Polytechnic University, Hong Kong, China. Also she was a visiting scholar in the University of Sydney from 2019 to 2020. She is currently a Professor at the School of Computer Science and Engineering, Northeastern University, Shenyang, China. Her current research interests include network security and big data. She is a member of the IEEE.

Yue Zong received the Ph.D. degree in communication and information systems from Northeastern University, Shenyang, China, in 2021. From 2016 to 2017, she was a visiting student with University of Bristol, Bristol, UK. She is currently the Postdoc in Power China Huadong Engineering Corporation Limited, Hangzhou, China. Her research interests include network virtualization and energy communication.

Phee Lep Yeoh (Senior Member, IEEE) received the B.E. degree with University Medal and the Ph.D. degree from the University of Sydney, Australia, in 2004 and 2012, respectively. From 2012 to 2016, he was a Lecturer at the University of Melbourne, Australia, and from 2016 to 2023, he was a Senior Lecturer at the University of Sydney, Australia. In 2023, he joined the School of Science, Technology, and Engineering at University of the Sunshine Coast in Queensland, Australia. Dr. Yeoh is a recipient of the 2020 University of Sydney Robinson Fellowship, the 2018 Alexander von Humboldt Research Fellowship for Experienced Researchers, and the 2014 Australian Research Council Discovery Early Career Researcher Award. He has also received best paper awards at IEEE PIMRC 2023, IEEE ICC 2014, and IEEE VTC Spring 2013.

Lei Guo (Senior Member, IEEE) received the Ph.D. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2006. He is currently a Full Professor with Chongqing University of Posts and Telecommunications, Chongqing, China. He has authored or coauthored more than 200 technical papers in international journals and conferences. He is an Editor for several international journals. His current research interests include communication networks, optical communications, and wireless communications.

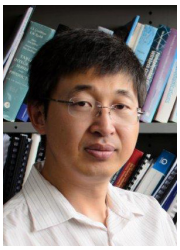
Branka Vucetic (Life Fellow, IEEE) received the Ph.D. degree from the University of Belgrade, Belgrade, Serbia, in 1982.

She is an ARC Laureate Fellow and Director of the Centre of Excellence for IoT and Telecommunications at the University of Sydney. Her current research work is in wireless networks and the Internet of Things. In the area of wireless networks, she works on communication system design for millimetre wave frequency bands. In the area of the Internet of Things, Vucetic works on providing wireless connectivity for mission critical applications.

Prof. Vucetic is a life Fellow of IEEE, the Australian Academy of Technological Sciences and Engineering and the Australian Academy of Science.

Trung Q. Duong (Fellow, IEEE) is a Canada Excellence Research Chair (CERC) and a Full Professor at Memorial University of Newfoundland, Canada. He is also the adjunct Chair Professor in Telecommunications at Queen's University Belfast, UK. He was a Distinguished Advisory Professor at Inje University, South Korea (2017-2019). He is an Adjunct Professor and the Director of Institute for AI and Big Data at Duy Tan University, Vietnam (2012-present), a Distinguished Professor at Thuyloi University, Vietnam (2023-2028) and a Visiting Professor (under Eminent Scholar program) at Kyung Hee University, South Korea (2023-2024). His research interests include quantum communications, wireless communications, signal processing, and machine learning.

Dr. Duong has served as an Editor/Guest Editor for the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, IEEE TRANSACTIONS ON COMMUNICATIONS, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE COMMUNICATIONS LETTERS, IEEE WIRELESS COMMUNICATIONS LETTERS, IEEE WIRELESS COMMUNICATIONS, IEEE COMMUNICATIONS MAGAZINES, and IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS. He received the Best Paper Award at the IEEE VTC-Spring 2013, IEEE ICC 2014, IEEE GLOBECOM 2016, 2019, 2022, IEEE DSP 2017, IWCMC 2019, 2023, and IEEE CAMAD 2023. He has received the two prestigious awards, including the Research Chair of the Royal Academy of Engineering (2021-2025) and the Royal Academy of Engineering Research Fellowship (2015-2020). He is the recipient of the prestigious Newton Prize 2017, and a Fellow of Asia-Pacific Artificial Intelligence Association (AAIA).



Yonghui Li (Fellow, IEEE) received his Ph.D. degree from Beijing University of Aeronautics and Astronautics, Beijing, China, in November 2002. From 1999 - 2003, he was affiliated with Linkair Communication Inc, where he held a position of project manager with responsibility for the design of physical layer solutions for the LAS-CDMA system. Since 2003, he has been with the Centre of Excellence in Telecommunications, the University of Sydney, Australia. He is now a Professor in School of Electrical and Information Engineering, University of Sydney. He is the recipient of the Australian Queen Elizabeth II Fellowship in 2008 and the Australian Future Fellowship in 2012. His current research interests are in the area of wireless communications, with a particular focus on MIMO, millimeter wave communications, machine to machine communications, coding techniques and cooperative communications. He holds a number of patents granted and pending in these fields.

Prof. Li is a Fellow of IEEE. He is now an editor for IEEE TRANSACTIONS ON COMMUNICATIONS and IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY. He also served as a guest editor for several special issues of IEEE journals, such as IEEE JSAC special issue on Millimeter Wave Communications. He received the best paper awards from IEEE International Conference on Communications (ICC) 2014, IEEE PIMRC 2017 and IEEE Wireless Days Conferences (WD) 2014.