Collaborative Learning Model for Cyberattack Detection Systems in IoT Industry 4.0

Tran Viet Khoa[1,2], Yuris Mulya Saputra[2], Dinh Thai Hoang[2], Nguyen Linh Trung[1],
Diep N. Nguyen[2], Nguyen Viet Ha[1], and Eryk Dutkiewicz[2]
[1] AVITECH, VNU University of Engineering and Technology, Vietnam National University, Hanoi, Vietnam
[2] School of Electrical and Data Engineering, University of Technology Sydney, Australia

***Abstract*—Although the development of IoT Industry 4.0 has brought breakthrough achievements in many sectors, e.g., manufacturing, healthcare, and agriculture, it also raises many security issues to human beings due to a huge of emerging cybersecurity threats recently. In this paper, we propose a novel collaborative learning-based intrusion detection system which can be efficiently implemented in IoT Industry 4.0. In the system under consideration, we develop smart "filters" which can be deployed at the IoT gateways to promptly detect and prevent cyberattacks. In particular, each filter uses the collected data in its network to train its cyberattack detection model based on the deep learning algorithm. After that, the trained model will be shared with other IoT gateways to improve the accuracy in detecting intrusions in the whole system. In this way, not only the detection accuracy is improved, but our proposed system also can significantly reduce the information disclosure as well as network traffic in exchanging data among the IoT gateways. Through thorough simulations on real datasets, we show that the performance obtained by our proposed method can outperform those of the conventional machine learning methods.**

Keywords- Cyberattack detection, Industry 4.0, IoT, federated learning, deep learning, and cybersecurity.

I. Introduction

The Industry 4.0 (known as the 4th industrial revolution) has emerged as one of the most innovative solutions for smart technology systems, e.g., smart factory, smart city, smart house, and smart office. The development of Industry 4.0 is expected to gain the greatest value by reducing manufacturing costs (47%), improving product quality (43%) and attaining operations agility (42%) [1]. In Germany, Industry 4.0 will contribute about 1% to annual GDP over the next ten years, creating as many as $390,000$ jobs, and adding €250 billion to manufacturing investment [2]. In Industry 4.0, IoT operates as a "bridge" to connect physical systems to the cyber world, and it enables manufacturing ecosystems driven by smart systems with autonomic self-properties, e.g., self-configuration, self-monitoring, and self-healing. With IoT, Industry 4.0 can achieve breakthrough achievements in many sectors, such as healthcare, food, and agriculture. For example, Industry 4.0 enables the food manufacturing sector to boost the operational productivity, reduce the production costs, and improve clean, safe and quality of products. However, when Industry 4.0 is connected to the cyber world, cybersecurity risks become a key concern due to open systems with IP addresses creating more avenues for cyberattacks. According to the 2016 Symantec Internet Security Threat Report, the manufacturing sector remained among the top 3 industries targeted by spear phishing attacks, suffering about 20% of all attacks. More seriously, for sensitive sectors, such as, healthcare and food industry, cybersecurity risks can cause serious effects to the human's lives. Therefore, countermeasures and risk mitigation solutions for cybercrime impacts are urgently in need.

To mitigate the damage caused by cyberattacks to the IoT Industry 4.0, it is essential to develop efficient solutions for early attack detection. For example, an attack detection approach based on the covariance matrix was proposed in [3]. In this approach, the attacks can be detected by discovering the correlation of various features in IP packet header captured from the network traffic. In [4], the authors introduced a classification technique using Kappa coefficient to detect and prevent Distributed Denial-of-Service (DDoS) attacks in the public cloud environment. In addition, the authors of [5] and [6] proposed to use autoencoder for anomaly detection to detect Botnet attack in the IoT environment. Nevertheless, these methods only can be implemented to detect some particular conventional attacks, e.g., DDoS and Botnet attacks and their performance in terms of accuracy is still limited. To address these issues, the authors in [7] developed a deep learning framework leveraging a deep belief network (DBN) that not only significantly improves the accuracy in detecting attacks, but also can detect a wide range of attacks, i.e., up to 38 types of attacks. The key idea of the deep learning approach is using a multi-layer neural network architecture to "learn" information from data many times over multiple layers. Thus, the learning quality of deep learning approaches can be greatly improved and outperform those of other conventional machine learning techniques. As a result, deep learning-based cyberattack detection systems have been received a lot of attention recently.

Despite possessing the outstanding advantages, the implementation of deep learning-based intrusion detection systems in IoT Industry 4.0 is facing several technical challenges. First, the IoT Industry 4.0 is a decentralized network with many subnetworks (SNs) deployed for various purposes, such as manufacturing, agriculture, and logistics. Each SN only controls a small set of IoT devices, and thus the data collected from each subset is usually insufficient to train the DBN for the cyberattack detection system. The insufficient data for training reduces seriously the accuracy of deep learning mechanism [8]. Sharing data among SNs may cause privacy concerns and network congestion due to a huge amount of data will be exchanged over the Internet. Second, SNs are usually managed by IoT gateways and/or edge nodes which

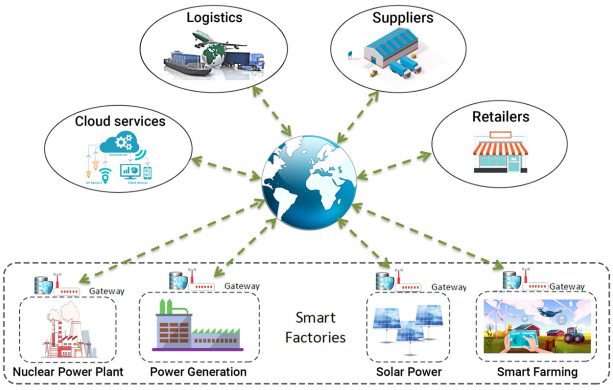

Fig. 1: IoT Industry 4.0 network architecture.

are limited by computing resources, and thus running deep learning algorithms with a huge dataset may not be efficient in a long-run.

In this paper, we propose a novel cooperative learning model which can be efficiently implemented on the cyberattack detection system in the IoT Industry 4.0 network. In particular, at each SN, we implement a smart "filter" on the IoT gateway which can promptly detect and prevent cyberattacks to its SN. The filter is developed based on a deep neural network (DNN), and its DNN is trained based on the data collected in its SN. To further improve the performance for the SNs, we propose a collaborative learning model in which the filters share their trained detection models with others instead of exchanging their real data. In this way, we can not only significantly enhance the accuracy in detecting attacks, but also boost the learning speed, reduce the network traffic, and highly protect data privacy for the SNs. Through simulation results on nine emerging IoT datasets and three conventional network datasets, we show that our proposed approach can improve the classification accuracy up to 50% and the communication overhead can reduce by 98% compared with those of other conventional machine learning techniques.

II. System Model

A. Network Architecture

Fig. 1 illustrates a general network architecture of the IoT Industry 4.0 network with multiple IoT subnetworks (SNs). In practice, each SN is deployed for a specific purpose, e.g., managing/monitoring nuclear nuclear power plant, power generation, solar power or smart farming. The IoT gateway in a SN serves as a "gate" to control and monitor all traffic in and out the SN. Each SN is controlled by a controller which can be located at the IoT gateway. The controller can implement a smart "filter", i.e., the deep neural network, in order to promptly detect and make decisions to protect its network. To facilitate the cyberattack detection process, the controller will store all data obtained in its network to a local database. This database will be updated regularly based on new incoming traffic, and it will be used to train the deep neural network for the cyberattack detection system inside its network.

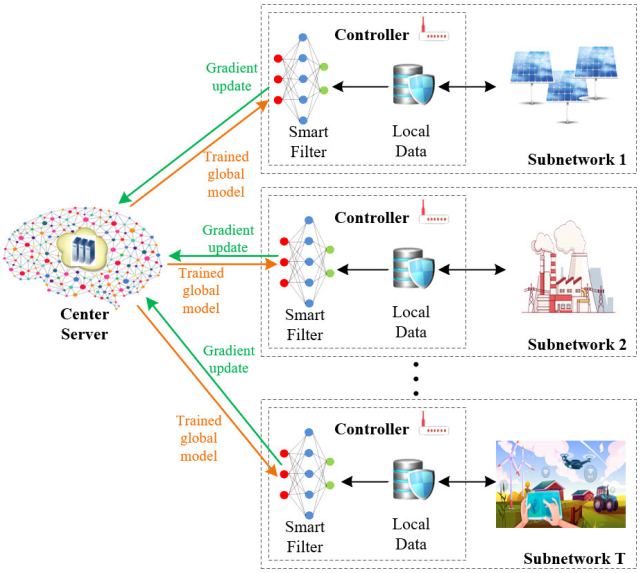

Fig. 2: Cyberattack detection system with collaborative learning model.

B. Cyberattack Detection System With Collaborative Learning Model.

To improve the efficiency of the cyberattack detection in the IoT Industry 4.0, we introduce a collaborative learning model with smart filters deployed at the IoT gateways as illustrated in Fig. 2. Each filter is controlled by its controller in its network and uses data in the local database to train its deep neural network. The trained model network will be then used to detect real-time cyberattacks. In the collaborative learning model, to exchange the trained model, a center server node (CS) will be used to collect the trained models from the filters and then gathering these models using the average gradient update algorithm to create the trained global model. After that the CS will send the trained global model to all the IoT gateways. Finally, based on the trained global model, each filter will update its local trained model. In this way, the filter of each SN can "learn the knowledge" from other filters without a need of sharing the raw dataset.

III. Collaborative Learning-based Cyberattack Detection Model

In this section, we propose two machine learning-based approaches which can be implemented in different scenarios in the IoT Industry 4.0 network. Specifically, we introduce classification-based and anomaly detection-based collaborative learning approaches to detect cyberattacks when the SNs in the IoT Industry 4.0 can only obtain labeled and unlabeled datasets, respectively.

A. Classification-based Collaborative Learning

This method is applicable to predict and identify the behavior of incoming packets for the cyberattack detection system. In particular, we use a deep learning approach utilizing deep belief network (DBN) to categorize the packets into

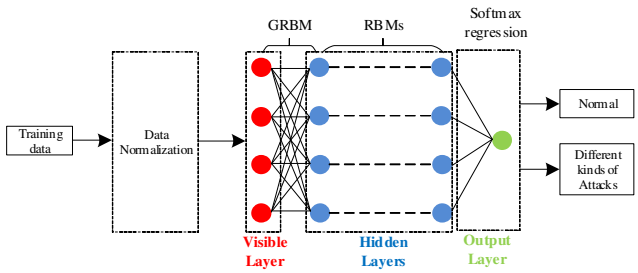

Fig. 3: Deep belief learning network architecture.

normal and abnormal behaviors [7]. As such, we can classify the packets into $M+1$ classes, where M refers to the types of attacks from the abnormal behavior. Consider $\mathbf{X} = \{\mathbf{X}_1, \ldots, \mathbf{X}_t, \ldots, \mathbf{X}_T\}$ as the training dataset containing the packets with normal and abnormal behaviors in the network, where T and $\mathbf{X}_t$ indicate the number of SNs and the training dataset at SN-t, respectively.

In the collaborative learning, each SN-t can learn its training dataset $\mathbf{X}_t$ locally. Alternatively, the CS only needs to collect the gradient information for the global model update without a need of downloading the training datasets from SNs. To predict the class of the incoming packets, each SN can perform the deep learning algorithm through visible and hidden layers of the DBN as illustrated in Fig. 3. Specifically, we first use a Gaussian Binary Restricted Boltzmann Machine (GRBM) [9] to convert the real training dataset at the input of visible layer into binary values at the first hidden layer. Let $\boldsymbol{\upsilon}_t = [\upsilon_1^t, \ldots, \upsilon_k^t, \ldots, \upsilon_K^t]$ and $\boldsymbol{\eta}_t = [\eta_1^t, \ldots, \eta_l^t, \ldots, \eta_L^t]$ denote the vectors of visible and hidden neurons of the visible and hidden layers at the SN-t, respectively. Here, K is the number of visible neurons and L is the number of hidden neurons in the GRBM. Then, the utility function of the GRBM at SN-t can be written as

$$\xi_t(\boldsymbol{\upsilon}_t, \boldsymbol{\eta}_t) = \sum_{k=1}^{K} \frac{(\upsilon_k^t - b_{1,k})^2}{2\gamma_{k,t}^2} - \sum_{k=1}^{K}\sum_{l=1}^{L} w_{k,l}\eta_l^t \frac{\upsilon_k^t}{\gamma_{k,t}} - \sum_{l=1}^{L} b_{2,l}\eta_l^t, \tag{1}$$

where $b_{1,k}$ and $b_{2,l}$ represent the global biases of visible and hidden neurons, respectively. Additionally, $w_{k,l}$ indicates the global weight between the visible and hidden neurons, and $\gamma_{k,t}$ specifies the standard deviation of visible neuron υ_k^t. Based on the Eq. (1), we can find the probability that a visible vector $\boldsymbol{\upsilon}_t$ at SN-t is used in the DBN as follows:

$$\rho_t(\boldsymbol{\upsilon}_t) = \frac{\sum_{\boldsymbol{\eta}_t} e^{-\xi_t(\boldsymbol{\upsilon}_t, \boldsymbol{\eta}_t)}}{\sum_{\boldsymbol{\upsilon}_t, \boldsymbol{\eta}_t} e^{-\xi_t(\boldsymbol{\upsilon}_t, \boldsymbol{\eta}_t)}}. \tag{2}$$

Then, we can obtain the local gradient of GRBM at SN-t for each epoch time τ, i.e., the time when all training dataset $\mathbf{X}_t$ at each SN-t has been observed, by

$$\nabla g_t^{(\tau)} = \sum_{k=1}^{K}\sum_{l=1}^{L} \nabla g_{t,k,l}^{(\tau)}, \tag{3}$$

where

$$\nabla g_{t,k,l}^{(\tau)} = \frac{\partial \log \rho_t(\boldsymbol{\upsilon}_t)}{\partial w_{k,l}} = \left\langle \frac{1}{\gamma_{k,t}} \upsilon_k^t \eta_l^t \right\rangle_{dataset} - \left\langle \frac{1}{\gamma_{k,t}} \upsilon_k^t \eta_l^t \right\rangle_{model}, \tag{4}$$

and $\langle . \rangle$ denotes the expectation value as described in [9].

Next, we execute deep learning process among the hidden layers using a Restricted Boltzmann Machine (RBM) [9]. In this case, the visible and hidden neurons have binary values, i.e., $[0, 1]$. Then, given K^* number of visible neurons and L^* number of hidden neurons, we can compute the utility function of the RBM at the SN-t as follows:

$$\xi_t^*(\boldsymbol{\upsilon}_t, \boldsymbol{\eta}_t) = -\sum_{k=1}^{K^*}\sum_{l=1}^{L^*} w_{k,l}\upsilon_k^t\eta_l^t - \sum_{k=1}^{K^*} b_{1,k}\upsilon_k^t - \sum_{l=1}^{L^*} b_{2,l}\eta_l^t. \tag{5}$$

Similar to GRBM, we can obtain the local gradient of RBM at SN-t for each epoch time τ by using Eq. (5) as follows:

$$\nabla g_t^{*(\tau)} = \sum_{k=1}^{K^*}\sum_{l=1}^{L^*} \nabla g_{t,k,l}^{(\tau)}, \tag{6}$$

where

$$\nabla g_{t,k,l}^{*(\tau)} = \left\langle \upsilon_k^t \eta_l^t \right\rangle_{dataset} - \left\langle \upsilon_k^t \eta_l^t \right\rangle_{model}. \tag{7}$$

From the last hidden layer of the DBN, each SN-t can obtain the output $\hat{\mathbf{X}}_t$ that will be used as the input of the softmax regression. In this case, the softmax regression is applied at the output of the DBN to classify the behaviors of the packets. Suppose $\mathbf{W}$ and $\mathbf{b}$ are the weight matrix and bias vector between the last hidden layer and the output layer, respectively. Then, the probability that Y belongs to class m and the prediction $\mathbf{Y}_t$ of packets' behaviors at SN-t are

$$\rho_t^\dagger(Y = m|\hat{\mathbf{X}}_t, \mathbf{W}, \mathbf{b}) = softmax_j(\mathbf{W}, \mathbf{b}) = \frac{e^{W_m \mathbf{X}_t + b_m}}{\sum_l e^{W_l \mathbf{X}_t + b_l}}, \tag{8}$$

and

$$\mathbf{Y}_t = \arg\max_m [p_t(Y = m|\hat{\mathbf{X}}_t, \mathbf{W}, \mathbf{b})], \forall m \in \{1, 2, \ldots, M+1\}, \tag{9}$$

respectively, where Y refers to an output prediction from $\mathbf{Y}_t$. Given Eq. (8), we can calculate the local gradient between the last hidden layer and the output layer as below

$$\nabla g_t^{\dagger(\tau)} = \frac{\partial \rho_t^\dagger(Y = m|\hat{\mathbf{X}}_t, \mathbf{W}, \mathbf{b})}{\partial \mathbf{W}}. \tag{10}$$

Upon obtaining $\nabla g_t^{(\tau)}$, $\nabla g_t^{*(\tau)}$, and $\nabla g_t^{\dagger(\tau)}$ for every τ, each SN-t sends the local gradients to the CS for global gradient aggregation as described by

$$\nabla g^{(\tau)} = \frac{1}{T}\sum_{t=1}^{T} \left(\nabla g_t^{(\tau)} + \nabla g_t^{*(\tau)} + \nabla g_t^{\dagger(\tau)} \right). \tag{11}$$

In this way, the CS works as a global model controller to accumulate the local gradients from the SNs synchronously, and then updates the global model before sending back to the SN-t, $\forall t \in \{1, \ldots, T\}$.. Specifically, suppose that $\phi^{(\tau)}$ is the

current global model at τ containing all weights of the DBN. The global model $\phi^{(\tau+1)}$ to learn $\mathbf{X}_t, \forall t \in \{1, \ldots, T\}$, for the next epoch time τ + 1 can be written as

$$\phi^{(\tau+1)} = \phi^{(\tau)} - \alpha \nabla g^{(\tau)}, \tag{12}$$

where α is the learning rate. The deep learning process continues and terminates when a convergence is reached or the the number of epoch time τ_{max} is achieved. As such, each SN can obtain the final global model ϕ^* containing the optimal weights for all layers (including weight matrix $\hat{\mathbf{W}}$). Then, we can find the final prediction $\hat{\mathbf{Y}}_t$ of packets' behaviors at SN-t as follows:

$$\hat{\mathbf{Y}}_t = \arg\max_m [p_t(Y = m | \hat{\mathbf{X}}_t, \hat{\mathbf{W}}, \mathbf{b})], \forall m \in \{1, \ldots, M+1\}. \tag{13}$$

Finally, we summarize the collaborative learning algorithm in Algorithm 1.

Algorithm 1 Collaborative learning based on classification algorithm

1: **while** $\tau \leq \tau_{max}$ or training process does not converge **do**
2: **for** $\forall t \in T$ **do**
3: Learn $\mathbf{X}_t$ to get $\mathbf{Y}_t$.
4: Calculate local gradients $\nabla g_t^{(\tau)}$, $\nabla g_t^{*(\tau)}$, $\nabla g_t^{\dagger(\tau)}$.
5: Send local gradients to the CS.
6: **end for**
7: the CS calculates the trained global model $\phi^{(\tau)}$.
8: $\tau = \tau + 1$.
9: Update the next global model $\phi^{(\tau+1)}$.
10: Send the updated global model $\phi^{(\tau+1)}$ back to T SNs.
11: **end while**
12: Predict $\hat{\mathbf{Y}}_t$ based on the training set $\mathbf{X}_t$ at each SN-t and optimal global model ϕ^*.

B. Anomaly Detection-based Collaborative Learning

This method is useful to detect anomaly in the case when the cyberattack detection system only has unlabeled dataset for the training deep neural network. In particular, we develop a collaborative learning model leveraging autoencoder network as illustrated in Fig. 4. Each SN-t generates the training dataset $\mathbf{X}_t$ containing packets with normal behavior only. Meanwhile, the testing dataset contains not only the packets with normal behavior, but also the packets with abnormal behavior coming from attack.

For the purpose of training process of autoencoder network, the training dataset $\mathbf{X}_t$ is separated into three dataset: $\mathbf{X}_{train}$, $\mathbf{X}_{opt}$, $\mathbf{X}_{test}$. To obtain accuracy prediction for the anomaly detection, the autoencoder network utilizes $\mathbf{X}_{train}$ to train the network and root mean square error (RMSE) loss function:

$$RMSE = \sqrt{\frac{1}{N} \sum_{n=1}^{N} \left(x - \hat{x}\right)^2}, \tag{14}$$

where N, x, and $\hat{x}$ are the number of samples, the actual packet behavior, and the predicted packet behavior, respectively. Unlike the classification method, the autoencoder network utilizes a gradient decent technique to re-generate the input data at the output layer while storing data properties, e.g., weights and biases, in the neural network. After that, the $\mathbf{X}_{opt}$ is used to create the margin for normal behavior identification:

$$Margin = \overline{RMSE}_{opt} + std(RMSE_{opt}), \tag{15}$$

where $\overline{RMSE}_{opt}$ is the mean of RMSE and $std(RMSE_{opt})$ is the standard deviation of RMSE with the $\mathbf{X}_{opt}$. Subsequently, the $\mathbf{X}_{test}$ is used to test the algorithm of training process. After the training process, the testing data with both normal and attack behavior is utilized for testing the anomaly detection. In testing process, the network behavior is considered attack when it has $RMSE > Margin$.

For the collaborative learning model using anomaly detection, we use the same mechanism as that of the classification method. In particular, each SN will train its model based on the anomaly detection algorithm, and then sends the trained model to the CS for global model update. After that the global model is sent back to the SNs for updates. This process is repeated until the algorithm converges or the maximum number of epoch time, τ_{max}, is achieved.

IV. Simulation Results

In this simulation, we use KDD [10], NSLKDD [11], UNSW-NB15 [12], and N-BaIoT [5], [6] datasets to evaluate the performance of the proposed approaches, i.e., collaborative learning model using classification and anomaly detection, by comparing to other baseline methods, i.e., centralized learning model for classification [7] and anomaly detection [5], [6], k-neigbours classifier, K-means, decision tree, multilayer perceptron, logistic regression, and support vector machine [13]. For the baseline methods, the CS first needs to collect datasets from all the SNs and then performs the machine learning

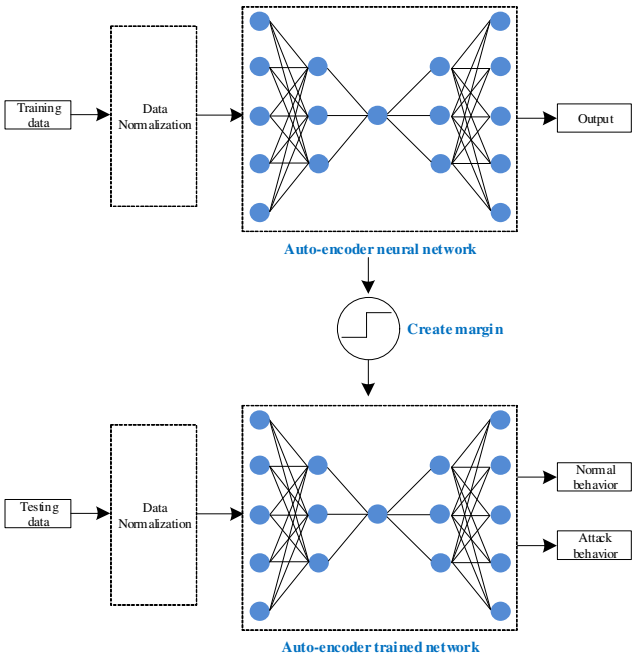

Fig. 4: Autoencoder network architecture.

TABLE I: The performance comparison of various machine learning methods over three traditional network datasets.

	KDD			NSL-KDD			UNSW		
	ACC	PPV	TPR	ACC	PPV	TPR	ACC	PPV	TPR
K Neighbours Classifier	88.56	77.19	71.39	94.31	77.42	71.52	96.85	94.12	92.13
K-means	82.78	84.96	56.95	87.05	74.01	35.23	86.19	89.16	65.47
Decision Tree	87.91	63.62	68.5	93.78	76.42	68.92	97.01	94.14	92.52
Multilayer Perceptron (MLP)	87.91	63.62	68.5	90.16	76.72	75.39	96.77	90.87	91.91
Logistic Regression	89.52	62.04	73.79	92.52	71.05	62.61	96.2	86.29	90.69
Support Vector Machine (SVM)	88.32	64.7	70.8	93.38	76.91	66.9	96.74	91.59	91.86
Centralized Deep Learning	97	94.26	92.52	90.86	80.68	77.15	95.67	82.48	78.33
Co-DL2	**97.52**	**94.71**	**93.79**	**93.99**	**85.16**	**84.98**	**95.6**	**82.62**	**78.01**
Co-DL3	**97.54**	**95.03**	**93.85**	**93.37**	**84.38**	**83.42**	**95.67**	**82.32**	**78.35**

algorithms to detect the normal and malicious packets. For the proposed method, we distribute the dataset into different SNs for the local training process.

A. Dataset Analysis

1) KDD dataset: The KDD dataset was built by DARPA Intrusion Detection Evaluation Program in 1998. This dataset includes 41 features, 24 types of attacks in the training dataset and 38 types of attacks in the testing dataset. The types of attacks are categorized into 4 groups including denial-of-service (DoS), attack from remote to local machine (R2L), unauthorized access to local administrator user (U2R), and probing attack.

2) NSL-KDD dataset: The NSL-KDD dataset [11] was built by cybersecurity group in the University of New Brunswick, Canada. Although this dataset contains the same properties of the KDD dataset, it eliminates many drawbacks of the KDD dataset including removing any duplicate samples in the dataset such that all records in both training and testing datasets are unique and providing better proportion of training and testing datasets.

3) UNSW-NB15 dataset: The UNSW-NB15 dataset [12] was created by Cyber Range Lab group of the Australian Centre for Cyber Security (ACCS). The dataset contains 49 features and 9 types of attacks with the class labels.

4) N-BaIoT dataset: The Network-based Detection of IoT Botnet Attacks Using Deep Autoencoders (N-BaIoT) dataset [5], [6] was developed by Yair Meidan from Ben-Gurion University of the Negev, Israel. This dataset contains the normal and malicious traffic from 9 IoT devices. Each dataset of the IoT device contains benign traffic and 10 types of attacks from Mirai and BASHLITE.

B. Evaluation Methods

As mentioned in [14], [15], the confusion matrix is typically used to evaluate the performance of system, especially machine learning system. We denote TP, TN, FP, and FN to be "True Positive", "True Negative", "False Positive", and "False Negative", respectively. Then, to evaluate the accuracy of the machine learning methods, we provide the following metrics:

1) Accuracy (ACC): This metric shows the correct detection of the learning process over the total traffic.

- *Classification*: If $M+1$ is the total number classes for normal and attack traffic, the ACC of class m is:

$$ACC_m = \frac{TP_m + TN_m}{TP_m + TN_m + FP_m + FN_m}.$$

Then, the ACC of the whole system can be calculated by averaging the ACC for $M+1$ classes:

$$ACC_{class} = \frac{1}{M+1} \sum_{m=1}^{M+1} \frac{TP_m + TN_m}{TP_m + TN_m + FP_m + FN_m}.$$

- *Anomaly detection*: If T_e is the total number of testing samples, then the ACC of anomaly detection is:

$$ACC_{anomaly} = \frac{1}{T_e} \sum_{t=1}^{T_e} 1(\hat{x} = x),$$

where $1(\hat{x} = x)$ is a Boolean function with value "1" indicating the correct prediction, and "0" otherwise.

2) Precision (PPV): This metric observes the real number of attack traffic over all predicted attack samples, i.e.,

$$PPV = \frac{TP}{TP + FP}.$$

3) Recall (TPR): This metric represents the proportion of correctly predicted attack traffic over all attack samples, i.e.,

$$TPR = \frac{TP}{TP + FN}.$$

Apart from the aforementioned metrics, we also analyze the complexity, i.e., the data transmission in the network, by comparing the learning time of all methods.

C. Performance Evaluation

In this section, we compare the performance of the proposed and baseline methods in terms of the accuracy, privacy, communication overhead, and learning time. For the collaborative learning-based methods, we distribute the dataset into T different SNs such as 2 SNs (Co-DL2) and 3 SNs (Co-DL3). Table I shows accuracy in detecting attacks between the proposed methods, i.e., Co-DL2 and Co-DL3, and other conventional learning methods. Generally, when we utilize traditional network datasets, the Co-DL3 can improve the ACC, PPV, and TPR performance up to 14.76%, 32.99%, and 49.75%, respectively, compared to the other results obtained by the conventional learning methods [7]. In this case, we can obtain the best performance using Co-DL3 when the

TABLE II: The performance comparison of various machine learning methods over nine emerging IoT datasets.

Id	IoT devices	Centralized Deep Learning			Co-DL2			Co-DL3		
		ACC	PPV	TPR	ACC	PPV	TPR	ACC	PPV	TPR
1	Danmini_Doorbell	89.56	99.54	79.5	**99.74**	**99.48**	**100**	**99.84**	**99.69**	**100**
2	Ecobee_Thermostat	98.08	96.32	100	**99.29**	**98.6**	**100**	**99.11**	**98.25**	**100**
3	Ennio_Doorbell	67.17	97.39	35.3	**67.53**	**98.21**	**35.52**	**67.27**	**97.42**	**34.96**
4	Philips_B120N10_Baby_Monitor	98.53	97.15	99.99	**98.64**	**97.38**	**99.98**	**98.96**	**97.97**	**100**
5	Provision_PT_737E_Security_Camera	85.83	98.96	72.42	**98.73**	**97.52**	**100**	**99.74**	**99.48**	**100**
6	Provision_PT_838_Security_Camera	86.89	99.62	74.06	**99.84**	**99.7**	**99.98**	**99.81**	**99.63**	**99.99**
7	Samsung_SNH_1011_N_Webcam	99.05	98.15	99.98	**98.83**	**97.71**	**100**	**98.86**	**97.78**	**99.98**
8	SimpleHome_XCS7_1002_WHT_Security_Camera	88.15	99.9	76.37	**99.39**	**98.84**	**99.97**	**99.56**	**97.2**	**99.98**
9	SimpleHome_XCS7_1003_WHT_Security_Camera	98.48	97.05	100	**98.43**	**96.99**	**100**	**98.41**	**96.99**	**100**

KDD dataset is used. The same trend can be observed for the Co-DL2. Although the Co-DL2 produces lower detection accuracy than that of the Co-DL3 by 1.5%, the Co-DL2 can still outperform other conventional learning methods. Then, we observe the anomaly detection using emerging IoT datasets in Table II. Compared to the centralized method, the proposed methods can increase the ACC, PPV and TPR by 13.91%, 0.53% and 27.58%, respectively.

In addition to the improvement of intrusion detection accuracy, the proposed methods can reduce the network traffic in the whole system significantly. Specifically, the proposed methods can reduce the network overhead by 98.5% compared with the conventional learning methods when KDD, NSL-KDD, UNSW-NB15, and N-BaIoT datasets are applied. The reason is that the SNs only need to transmit the small-size trained models, i.e., local gradient information, instead of sending the whole dataset to the CS. Furthermore, this trend aligns with the privacy disclosure reduction as the the SNs train the dataset locally. In this way, all the SNs can collaborate with each other through the CS without revealing the private information.

Next, we compare the learning speed performance of the learning methods in Fig. 5. It can be observed that the learning speed of Co-DL2 method is 30% faster than that of the centralized method. Additionally, when we apply the Co-DL3, we can further increase the learning speed by 40% compared with the centralized method. This is because, in the proposed methods, we can distribute the dataset to different SNs with respect to the number of SNs in the network. Consequently, each SN can perform the deep learning algorithm using smaller number of samples efficiently.

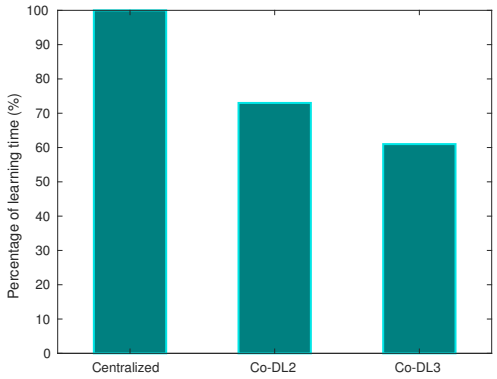

Fig. 5: Learning speed comparison for various methods.

V. Conclusion

In this paper, we have proposed the novel intrusion detection system based on the collaborative learning model in IoT Industry 4.0. Specifically, we have designed the smart "filters" at the IoT gateways to train the collected data locally using the deep learning algorithm, aiming at detecting and preventing cyberattacks. To significantly enhance the accuracy in detecting intrusions, and reduce the network traffic as well as the information disclosure, we have proposed a collaborative learning model which allows the filter to learn information from others through exchanging the trained models only. Through extensive simulations, we have demonstrated that the performance of the proposed method can outperform other conventional machine learning methods on the real dataset in terms of the detection accuracy, network traffic, privacy disclosure, and learning speed.

VI. Acknowledgement

This work is the output of the ASEAN IVO http://www.nict.go.jp/en/asean_ivo/index.html project Cyber-Attack Detection and Information Security for Industry 4.0 and financially supported by NICT http://www.nict.go.jp/en/index.html

References

[1] The Boston Consulting Group, "Sprinting to Value in Industry 4.0". Available Online: http://r3ilab.fr/wp-content/uploads/2017/01/BCG-Sprinting-to-Value-in-Industry-4-0-Dec-2016.pdf.

[2] The Boston Consulting Group, "Industry 4.0: The future of productivity and growth in manufacturing industries". Available Online: https://www.zvw.de/media.media.72e472fb-1698-4a15-8858-344351c8902f.original.pdf.

[3] M. N. Ismail, A. Aborujilah, S. Musa, and A. Shahzad, "Detecting flooding based DoS attack in cloud computing environment using covariance matrix approach," in *Proceedings of the 7th international conference on ubiquitous information management and communication*. ACM, 2013, pp. 36:1–36:6.

[4] A. Sahi, D. Lai, Y. Li, and M. Diykh, "An efficient DDoS TCP flood attack detection and prevention system in a cloud environment," *IEEE Access*, vol. 5, pp. 6036–6048, April 2017.

[5] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-BaIoT-Network-based detection of IoT botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, July 2018.

[6] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: an ensemble of autoencoders for online network intrusion detection," *arXiv preprint arXiv:1802.09089*, 2018.

[7] K. K. Nguyen, D. T. Hoang, D. Niyato, P. Wang, D. Nguyen, and E. Dutkiewicz, "Cyberattack detection in mobile cloud computing: A deep learning approach," in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, April 2018, pp. 1–6.
[8] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT Press, 2016.
[9] G. E. Hinton, "A practical guide to training restricted Boltzmann machines," in *Neural networks: Tricks of the trade*. Springer, 2012.
[10] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.
[11] "University of New Brunswick," https://www.unb.ca/cic/datasets/nsl.html.
[12] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *2015 military communications and information systems conference (MilCIS)*, Nov 2015, pp. 1–6.
[13] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *Journal of Internet Services and Applications*, vol. 9, no. 1, pp. 1–99, Jun 2018.
[14] T. Fawcett, "An introduction to ROC analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, June 2006.
[15] D. M. Powers, "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation," *Journal of Machine Learning Technologies*, pp. 37–63, 2011.