Federated Semi-Supervised Learning for Attack Detection in Industrial Internet of Things

Ons Aouedi, Kandaraj Piamrat, Guillaume Muller, Kamal Singh

Federated Semi-Supervised Learning for Attack Detection in Industrial Internet of Things

Ons Aouedi*, Kandaraj Piamrat*, Guillaume Muller+, and Kamal Singh+
*LS2N, Universite de Nantes, BP 92208, 44322 Nantes Cedex 3, France
{firstname.lastname}@ls2n.fr
+Univ Jean Monnet, IOGS, CNRS, UMR 5516, LaHC, F - 42023 Saint-Etienne, France
{firstname.lastname}@univ-st-etienne.fr

***Abstract*—Security has become a critical issue for Industry 4.0 due to different emerging cyber-security threats. Recently, many Deep Learning (DL) approaches have focused on intrusion detection. However, such approaches often require sending data to a central entity. This in turn raises concerns related to privacy, efficiency, and latency. Despite the huge amount of data generated by the Internet of Things (IoT) devices in Industry 4.0, it is difficult to get labeled data, because data labeling is costly and time-consuming. This poses many challenges for several DL approaches, which require labeled data. In order to deal with these issues, new approaches should be adopted. This paper proposes a novel federated semi-supervised learning scheme, that takes advantage of both unlabeled and labeled data in a federated way. First, an AutoEncoder (AE) is trained on each device (using unlabeled local/private data) to learn the representative and low-dimensional features. Then, a cloud server aggregates these models into a global AE using Federated Learning (FL). Finally, the cloud server composes a supervised neural network, by adding fully connected layers (FCN) to the global encoder (the first part of the global AE) and trains the resulting model using publicly available labeled data. Extensive case studies on two real-world industrial datasets demonstrate that our model: (a) ensures that no local private data is exchanged; (b) detects attacks with high classification performance, (c) works even when only a few amounts of labeled data are available; (d) has low communication overhead.**

***Index Terms*—Data privacy, federated learning, machine learning, deep learning, semi-supervised learning, intrusion detection.**

I. Introduction

Recent advances in communication and Internet of Things (IoT) technology enable the realization of the Industrial Internet of Things (IIoT). IIoT refers to the networking and connection of production equipment, instruments, products, and sensors, with the industrial and manufacturing applications [1]. In IIoT, a large number of smart devices and sensors act in the background to collect the environment and user data. However, these devices are vulnerable to several cyber-attacks [2] where the attackers can intercept and analyze some sensitive data [3]. As a consequence, there is an urgent need to design efficient attack detection mechanisms. As a counter measure, more intelligent methods need to be designed and deployed in the network. Recently, Machine/Deep Learning (ML/DL) models have shown potential due to their ability to analyze the traffic and extract knowledge out of it. ML/DL models can automatically diagnose and detect attacks using flow and packet-based features. In fact, building DL models consists of three steps: i) data capture and labeling; ii) data pre-processing, and iii) model training. The first step requires that data are manually labeled after being captured, which is a highly expensive and time-consuming process. The second and third steps require the data owners to send their private data to a central entity for pre-processing and model training. This raises privacy issues as confidential data might need to be shared in the process.

Data labeling and unsupervised learning. Semi-supervised learning can exploit abundant unlabeled data in combination with a small amount of labeled data, thus has received a lot of attention recently to solve the problem of labeling data. The unsupervised learning captures the most relevant features from the unlabeled data in order to provide a better representation of the data than the initial raw data (i.e., input data) itself. After capturing these features, the supervised training uses the labeled data to fine-tune model parameters and constructs the final model. The final model can then be used for example to classify the traffic as either benign or an attack.

Data privacy and Federated Learning (FL). In the Industry 4.0 environment, the datasets are distributed across the IIoT devices, where the data of each device is limited in diversity and quantity. Moreover, due to privacy concerns, the devices may not be willing to share their data with a central entity. Besides, privacy concerns, collecting the data to a central entity could lead to significant communication overhead as well as introduce network congestion [1]. All this makes it impractical to collect raw data from different IIoT devices. Solving the above issues requires a collaborative approach where the intrusion detection system is built while leaving the data at the location of its production. Federated Learning solves exactly this problem. FL does not need to move the data to a central entity. It is an iterative learning process where the model performance can be improved during each communication round [4]. More specifically, the IIoT devices (e.g., industrial machines or robots) using FL can train a model locally. Then, they can just send the model parameters instead of the raw data to a central entity, which can aggregate the different received models.

In our work, we address both the problems of data labeling and privacy at the same time, by combining semi-supervised learning with FL. This allows the IIoT devices to participate in the training process without labeling and sending their data to a central entity and in turn increase the scalability and robustness of IIoT applications as well as overcome data limitation problems and privacy concerns.

A. Related work

In this section, we first present some representative solutions of recently proposed approaches for intrusion detection based on conventional ML/DL models (Section I-A1). Then, we present the recent achievements of state-of-the-art approaches that proposed intrusion detection systems using FL-based architecture (Section I-A2).

1) Conventional Machine/Deep Learning models: ML/DL models offer a way to detect attacks efficiently, as they can find useful patterns in data in a reasonable time. This ability to generalize from examples also enables them to better adapt to new kinds of attacks than handcrafted Intrusion Detection Systems (IDS). In this context, Ge et al. [5] proposed a novel intrusion detection approach for IoT systems using a DL model. Specifically, a feed-forward neural network (FNN) has been used for binary and multi-class classification for four categories of attacks, which are reconnaissance, DoS, DDoS, and information theft attacks. The experimental results demonstrated that the proposed method outperforms Support Vector Machine (SVM) in terms of accuracy and training time.

At the same time, several IDS based on the distributed architecture of fog computing nodes and ML/DL models were proposed, for example, Kumar et al. [6] proposed a novel ensemble learning for intrusion detection using fog nodes. It is composed of two levels. In the first level, three supervised models are trained on the initial data. Then, in the second level, Random Forests (RF) have been used to combine the output of the first level and provide the final classification (”attack” or ”benign”).

However, as new types of attacks emerge every day, this brings new challenges that need to be tackled. In fact, labeling data is often difficult and time-consuming. Therefore, the researchers started to use a semi-supervised learning model where they can exploit both labeled and unlabeled data. In this context, Aamir et al. [7] proposed a semi-supervised model for DDoS attack detection. In the first step, principal component analysis (PCA) was used for data reduction and feature extraction. In the second step, a clustering algorithm was used to label the data based on their cluster. Finally, after label assignment, several supervised models are applied for training and attack detection. The experimental results show that the RF model is more accurate than K-Nearest Neighbors (KNN) and SVM. In the same context, Hara et al. [8] proposed an IDS system using a semi-supervised learning model. They used an adversarial auto-encoder (AAE) to extract the relevant features. The experimental results demonstrate that the proposed model can achieve high accuracy with a minimal number of labeled data as compared to deep neural networks. To evaluate the performance of their solution, the authors used `NSL-KDD` dataset, which is quite an old dataset and can miss modern network behaviors.

All the above works are based on the centralized training system where data are collected and processed in a single central server. This type of solution can be computationally expensive, time-consuming, prone to threats, and problems of information security and can cause leakage of confidential data. In contrast, through the use of FL, our model tries to detect attacks without breaching data privacy as well as reducing the communication and storage overhead as much as possible by sending only model parameters to a central server instead of the raw private data.

TABLE I: Summary of existing works.

Schemes	**Characteristics**		
	Federated learning	Semi-supervised learning	Communication overhead minimisation
[9] Ensemble FL-based attacks detection (GRUs, RF)	✓		
[10] MV-FLID: ensemble FL-based intrusion detection (FNN, RF)	✓		
[8] Semi-supervised model-based intrusion detection (AAE)		✓	
[11] Ensemble FL based attacks detection (CNN, GRUs)	✓		
[12] DL-based intrusion detection (GRUs)	✓		
[13] Federated transfer learning based intrusion detection (CNN)	✓		
[7] Semi-supervised DDoS attack detection (PCA, K-means, RF)		✓	
[14] FL-based attacks detection (DNN)	✓		✓
[15] FL-based attacks detection (DNN, CNN, RNN)	✓		
[16] FL-based attacks detection (DNN)	✓		
[17] FL-based Android malware detection (GAN)	✓		
Our model (Federated semi-supervised attacks detection (AE, FCN))	✓	✓	✓

2) Federated Learning: FL was proposed to ensure data privacy during ML/DL model training. For instance, Nguyen et al. [12] introduced an FL system for detecting compromised IoT devices, called `DÏoT`. Gated Recurrent Units (GRUs) have been used as a DL model. This is the first system that deployed FL for IDS. It consists of two components, which are security gateways and IoT security services. Security Gateways use the local data to train the local models and IoT security service aggregates the local models into a global model.

Taheri et al. [17] introduced an FL approach to detect Android malware in IIoT, called `Fed-IIoT`. Specifically, `Fed-IIoT` uses two generative adversarial networks (GAN) models to generate adversarial data and inject them into the dataset. The server uses a GAN in order to detect a malicious model and to delete the poisoned data. `Fed-IIoT` has been tested on three IoT datasets and the results confirm the higher attack detection performance of their approach.

Mothukuri et al. [9] proposed an ensemble FL-based attack detection and classification in IoT networks. The authors combine RF and GRUs models to construct their ensemble. In other words, they used RF in order to combine the predictions from the GRUs model to further improve the classification performance of the FL approach. The experimental results demonstrate a minimized error rate in predicting attacks and a reduced number of false alarms in comparison to the centralized ML approaches.

Similarly, Attota et al. [10] proposed an ensemble FL-based intrusion detection, called `MV-FLID`. Specifically, the authors have developed three FNN models for three views (i.e., Biflow View, Packet View, and Uniflow View), and the outcomes of these models are sent to an ensembler model (RF), which combines the predictions of these models and classifies the instances. The results show that the FL approach can outperform the Non-FL one. Li et al. [11] proposed a federated DL scheme in cyber-physical systems (CPS), called `DeepFed`. A combination of Convolutional Neural Networks (CNN) and GRUs has been used for intrusion detection. The experimental results on a real industrial dataset demonstrate the high accuracy of `DeepFed` as compared to some other FL approaches. Popoola et al. [14] proposed FL model for zero-day attack detection in IoT edge devices. The experimental results demonstrate that FL outperforms the state-of-the-art methods in terms of data privacy, communication overhead, and memory storage space as well as the attack detection accuracy. In the same direction, Fan et al. [13] proposed another model for IDS in 5G IoT network using federated transfer learning, called `IoTDefender`. They use FL to aggregate information without data sharing and transfer learning to ensure a personalized model for each IoT network. In addition, Ferrag et al. [15] presented a comprehensive survey including experimental analysis of FL approaches for cyber-security, in the IoT domain, using several datasets. Finally, Sarhan et al. [16] proposed a collaborative FL approach for IDS using a DL-based model. Their results demonstrated that FL achieved similar performance with that of the centralized approach in the binary and multi-classification scenarios.

The summary of the above works demonstrates the effectiveness of FL for IDS however as shown in Table I none of the existing works exploits the unlabeled data during the FL training process. The integration of both FL and semi-supervised learning for intrusion detection is what currently lacks in the state of the art and that our model addresses.

B. Motivations and Key Contributions

The motivations behind the use of the federated semi-supervised learning model are as follows. Although semi-supervised learning has been used in the literature for IDS, it is often limited by the scale and hence can not generalize very well. Secondly, user data protection is recently accentuated by several international regulatory policies, which restrict data access and protect medical data privacy. For example, the General Data Protection Regulation in European Union (GDPR)[1] completely redefines the data management policy. Thirdly, labeling data can be costly and time-consuming, whereas using unlabeled data can also provide valuable information for the model and in turn boost the model classification performance. Therefore, the integration of FL and semi-supervised learning for the attack detection task is a promising direction. However, the large body of literature presented above has not explored the combination of FL and semi-supervised learning for attack detection.

To address these limitations, we propose an FL-based semi-supervised model for ensuring privacy as well as for taking advantage of the unlabeled and labeled data in the IIoT environment. More specifically, we train a model using only a small amount of labeled data combined with more abundant unlabeled data. Additionally, our proposal can enhance the model performance through the collaboration of data sources which are the distributed IIoT devices. The main contributions of our research are as follows:

- We propose a federated semi-supervised approach for intrusion detection, by integrating FL and semi-supervised learning in one model which is used by the distributed IIoT devices.
- We use the AE as the unsupervised model at the IIoT devices in order to reduce the size of local models and to decrease the communication overhead.
- We use the cloud server as follows. It first generates the global AE using FL. Next, the cloud server adds additional layers to the global encoder block and trains the resulting neural network further using supervised learning with publicly available labeled data.
- We further deploy a joint-announcement protocol [18], which uses clients' random selection in order to decrease the communication overhead.
- We perform an extensive simulation using two industrial datasets, which are benchmark datasets, as well as we compare our approach against some state-of-the-art approaches.

C. Paper Structure

The rest of this paper is organized as follows. Section II-A, firstly presents essential background, then Section II-B introduces our FL-based semi-supervised algorithm for intrusion

[1]https://gdpr-info.eu/issues/data-protection-officer/

and attack classification. Experimental settings and results as well as the dataset are presented in Section III. Discussion and analysis of the results and the future directions are provided in Section IV. Finally, the conclusion is given in Section V.

II. Proposed solution

This section presents the main components of our system as well as the architecture of the FL-based semi-supervised learning.

A. Components

1) AutoEncoder: AE is an unsupervised neural network model. It consists of two blocks, (i) *encoder* and (ii) *decoder*. The *encoder* takes input data and maps it to a hidden representation (latent representation), called *code*. Then, the *decoder* uses the hidden representation to reconstruct the input. The main purpose of the AE is dimensionality reduction of the input data by extracting important features as the latent variable vector. For more details about AE, please refer to [19].

AE structure with single hidden layer is formulated as:

$$encoder : Z = f(W_1 X + b_1) \tag{1}$$

$$decoder : X' = f(W_2 Z + b_2) \tag{2}$$

where $X = (x_1, x_2, \ldots, x_n)$ is the input vector, and $Z = (z_1, z_2, \ldots, z_m)$ is the vector known as latent space which in turn is extracted from the input X, $X' = (x'_1, x'_2, \ldots, x'_n)$ is the output reconstruction of the input X, where n is the dimension of the input vector and m is the number of code units. W_1 and b_1 are the weight matrix and bias between the input layer and the second layer (i.e., code). W_2 and b_2 are the weight matrix and bias between the second and the output layer; $f(.)$ is the activation function.

The difference between X and X' is usually called the reconstruction error RE, which is represented in the form of a cost function that the model tries to reduce during the training process. The cost function of the AE is computed using Equation 3, where the parameter set is denoted by $\theta = \{W_1, b_1, W_2, b_2\}$.

$$J(\theta) = \sum_{i=1}^{n} RE(x_i, x'_i) \tag{3}$$

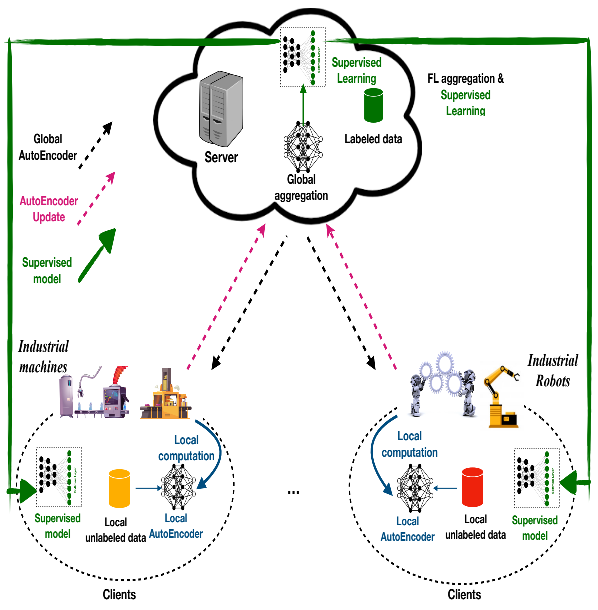
2) Federated Learning: FL attempts to answer the following main question: *Can we train the model without the need to transfer data over to a central location?* Within the FL concept, the data is maintained where it is generated and no raw data gets exchanged. In other words, FL is a distributed machine learning framework where the data entities/clients collaborate to jointly learn a global model (e.g. intrusion detection) without sacrificing the privacy of the end-users as much as possible. Given K clients $\{C_1, \ldots, C_K\}$ and their respective local data $\{D_1, \ldots, D_K\}$, the FL framework builds a model through the collaboration of several clients $n \leq K$ without exposing their data to others. First, the central server specifies the hyperparameters of the global model and the training process. Then, it disseminates the initial global model to several clients. Based on this model, each client updates the global model locally for a selected number of epochs using its own data. Once the local training is finished, these clients send their own model updates to the central server that aggregates these updates and computes the global model without exchanging data. These steps are repeated until the global model achieves satisfactory accuracy. The aim of the global aggregation is to aggregate the local models of different clients. For this task, the Federated Averaging (FedAvg) algorithm [4] is the most simple and frequently used one; hence, it will be used with our model. By making devices exchange model parameters rather than their raw local private data, the FL process limits the communication and storage overhead.

B. Methodology

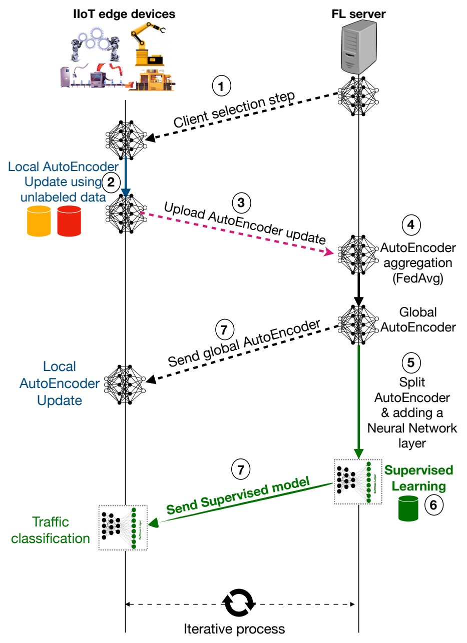
Our objective is to train a semi-supervised model using only a small amount of labeled data while preserving data privacy. Figure 1 and Algorithm 1 present the architecture and the main idea of our model. Table II presents the notations used in Algorithm 1.

Our model consists of two parts: the clients' side and the server-side. The clients perform the model pre-training using their *unlabeled* data and the server fine-tunes the global parameters using its limited *labeled* data (Figure 1 (a)). In **step 1** in Figure 1 (b), the server selects a random subset of clients, $n \leq K$, that will participate in the learning process and send them the initial AE model. Then, on the clients' side (IIoT devices), the AE model is trained for a selected number of epochs using the clients' unlabeled data with the objective of reducing the reconstruction error **(step 2)**. Here, the total unlabeled data are randomly distributed across the clients. In an IIoT context, clients are often far from human reach, thus accessing them to label their data is a difficult and impractical task. Thus, to be more realistic, in this work we consider that the client data are fully unlabeled. Then, the clients send their local models to the cloud server for global aggregation using the FedAvg algorithm **(step 3)**. On the server-side, the AE is aggregated **(step 4)**, then the decoder is removed and a Fully Connected Network (FCN) layer is attached to the encoder layers in order to fine-tune the model parameters for the supervised learning using a limited labeled data located on the server **(step 5 and 6)**. Specifically, the server uses its limited labeled data for supervised learning, and thus unlike the classical FL, in our case, the server is not only used for the model aggregation but also for supervised learning. Finally, the server sends back the global AE to the clients for a further update as well as the trained supervised model for inference (classification of the attacks) on the network data of the devices **(step 7)**. It is important to note that (i) no raw data is exchanged between the clients and the central server, and (ii) the supervised model is trained through domain-specific public datasets or laboratory data located on the server without privacy concerns.

In our federated semi-supervised learning model, the computational complexity is related to the computations at each IIoT device or client (which performs the unsupervised learning) and at the FL server (supervised learning), plus their com-



(a) Federated semi-supervised architecture



(b) Communication process

Fig. 1: The network architecture and communication process of our Federated semi-supervised proposal for IIoT.

Algorithm 1: Learning procedure.

```
1:  Input: Public labeled Dataset $D^l = \{x_i, y_i\}$
    $(i = 1, 2, \ldots, n;\ x_i \in X;\ y_i \in Y)$; Private unlabeled
    dataset $D^u_k = \{x_i\}$ $(k = 1, 2, \ldots, K;\ x_i \in X_k)$, $R$, $E_c$,
    $E_s$, $K$, $r_k$
    /* --- Server side ---                                  */
2:  Send initial global model $\theta = \theta_0$ to clients
3:  for $i = 1$ to $R$ do
4:    $n$= $r_k$ * $K$
5:    for $j = 1$ to $n$ do in parallel
    /* --- Client side ---                                  */
6:      for $e_c = 1$ to $E_c$ do
7:        Update local AE parameters $\theta_j$ using $D^u_j$
8:      end for
9:      Send updated $\theta_j$ to the Server
10:   end for
    /* --- Server side ---                                  */
11:   Aggregate $\{\theta\}_{j=1 \ldots n}$ with FedAvg into $\theta$
12:   Extract encoder
13:   Concatenate encoder and FCNLayer into H
14:   for $e_s = 1$ to $E_s$ do
15:     Train H using $D^l$
16:   end for
17:   Send models $\theta$ and H to the clients
18: end for
```

munications. The global aggregation results in insignificant computational costs. Thus, here we focus on analyzing the computational complexity related to learning at each device and the FL server. We assume that L_{AE} and L_H are the number of layers of the AE model and supervised model, respectively. For simplicity, we assume that each layer has N neurons. The computational complexity comes from back-propagation algorithm. To calculate new weights for each layer, we require matrix multiplications for applying gradients, which is the most complex part of the process. Assuming simplest matrix multiplication algorithm, the complexity of multiplying two matrices (with N rows and $size(D^u_k)$ columns into $size(D^u_k)$ columns and N rows) is $\mathcal{O}(N \times size(D^u_k) \times N)$. Considering the training epochs and the number of layers, the computational complexity of the AE at each client becomes $\mathcal{O}(size(D^u_k) \times E_c \times L_{AE} \times N^2)$. Similarly, the complexity of the supervised learning on the FL server is $\mathcal{O}(size(D^l) \times E_s \times L_H \times N^2)$.

TABLE II: List of notations used in our model.

Notation	Meaning
K	Total number of clients
r_k	fraction of the clients randomly selected from K
n	The number of clients selected at each round
E_c	Local clients epochs
E_s	Local server epochs
R	Total number of rounds
H	Supervised model
θ_0	Initial AE parameters

III. Experiments and Performance Evaluations

In this section, we evaluate the performance of our model through extensive experiments and discuss the results.

A. Data Description

To validate the effectiveness of our proposition, we used a gas pipeline SCADA system dataset, which is a benchmark dataset for security research [20]. It was released by the Mississippi State University in 2014. This dataset consists of 26 features and 1 label, the label contains eight possible values, benign and seven different types of attacks. The possible values for the label are presented in Table III. In addition, as this dataset consists of different features with values in different scales, we have standardized our data.

TABLE III: Dataset description.

Label	Description	# Total observations
Benign	Normal traffic	61,156
NMRI	Naive malicious response injection	2,763
CMRI	Complex malicious response injection	15,466
MSCI	Malicious state command injection	782
MPCI	Malicious parameter command injection	7,637
MFCI	Malicious function command injection	573
DoS	Denial of service	1,837
Rec	Reconnaissance	6,805

B. Experimental Setup

In this study, we split the dataset into two subsets: train (80%) and test (20%). We use Python as a programming language and `Scikit-learn` for the conventional models and `PyTorch` for DL models. All experiments are run on a four-core Intel® Core™ i7-6700 CPU@3.40GHz processor, and 32GB of RAM. Table IV summarizes the model parameters and their selected settings in our simulations. The simulation of a partially-labeled dataset from this fully labeled data has been done by randomly selecting rows from the training set and removing their labels. It is important to note that during this experiment, we fixed the amount of labeled data and vary only the amount of unlabeled data. As the dataset is imbalanced (there are much more examples of some types of attacks than others), we use the *F1-score* (i.e., the harmonic mean of precision and recall) as an evaluation metric, together with the *accurracy*). In addition, in our experiments, we compute the average of the evaluation metric from five runs.

TABLE IV: Implementation parameters.

Deep Learning	
Deep learning tool	PyTorch
DL algorithms	Autoencoder (AE) Fully Connected Layer (FCN)
AE hidden layers	8
encoder neurons	[Input, 20, 15, 10]
FCN layers	1
Activation functions	ReLu (AE), Softmax (FCN)
Optimizer	Adam
Learning rate	0.001
Batch size	64
Dropout	0.01
Federated Learning	
FL server	1
Nb clients	100
Clients used in federated updates	10%
AE epochs (client)	20
FCN epochs (server)	100
Communication round	18

C. Performance under different factors

In this section, we study the performance (in terms of accuracy, F1-score, and communication overhead) of the proposed model under different factors, which are: fraction of the selected clients r_k, local client epochs E_c, communication rounds R, and amount of unlabeled data.

1) Impact of the number of selected clients: Here, we explore the impact of different numbers of clients on the classification performance of our model, while holding K fixed (K= 100). Before training, we distribute the unlabeled data into equally one hundred parts, meaning that each client gets $nb\_rows(D^u)/K$ observations randomly. Since in real-time situations, the probability of client failures is significant [21] we use the Joint-Announcement Protocol (JAP) to avoid the problem of client failure and communication overhead. Given the ratio of clients (r_k), JAP selects randomly the clients that participate in the i^{th} training round [18]. As shown in Table V we evaluate the performance of our classifier under different number of client n, where $n \in \{10, 30, 100\}$.

With $n = 100$ (i.e., the total number of clients) the classification performance of our model degrades little. Our model gets the best results when $n = 10$. This may be attributed to the fact that a large number of clients can increase the diversity of the AE model across the clients and degrade the classification performance.

TABLE V: F1-score for various numbers of clients.

C_k	**10**	**30**	**100**
F1-score	**88.14% ±0.16**	88.07% ± 0.07	87.91% ± 0.24

Further, we compare the communication overhead with and without JAP. Here we set the number of clients to $n = 10$ using JAP and $n = 100$ without JAP. As present in figure 2, we can notice that using JAP can reduce the communication overhead. Specifically, in every communication round, JAP with $n = 10$ reduces the overhead by **90%**. This demonstrates the critical impact of the clients' number on the FL system.

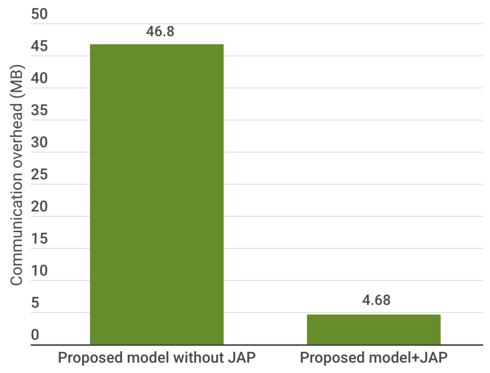


Fig. 2: Overhead of the proposed model without/with JAP.

2) Impact of the local epochs: In this subsection, we study the performance of our model in terms of F1-score and communication overhead using different values for the clients' local epochs E_c, used during the AE training (client level). As shown in Table VI, increasing the number of $E_c = 20$ improves the performance of our model. This may be attributed to the fact that increasing the local epoch of the AE helps to find more relevant features which, in turn, boosts the performance of the supervised model. However, increasing this parameter also increases the training time.

Moreover, as seen from Figure 3 the local epoch does not only impact the classification performance of the model but

TABLE VI: F1-score for various numbers of epochs on the client-side.

E_c	**20**	5	1
F1-score	**88.14% $\pm$ 0.16**	88.01% $\pm$ 0.18	87% $\pm$ 0.17

also the communication overhead. Specifically, here we show the communication overhead of our model under different frequency updates. It can be seen that increasing the number of local epochs in our model can decrease the communication overhead of the FL model. For example, when $E_c = 20$, the clients send only 1 time their model to the sever. However, with $E_c = 5$, and $E_c = 1$, they send their model to the server 4 and 20 times, respectively. Therefore, since the sensors in IIoT systems generate large volumes of data, increasing the local epochs improves the attack classification task at the same time it decreases the communication overhead; hence, improving the network overall performance.

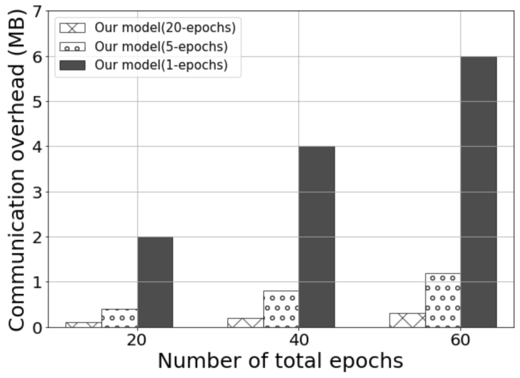


Fig. 3: Comparison in terms of communication overhead.

3) Impact of communication rounds: Here, we evaluate the classification performance (accuracy and F1-score) of our model under the different numbers of communication rounds R. As presented in Figure 4, the performance of the model increases with more interaction/round between the server and the clients. Also, it can be seen that the model converges quickly and its accuracy starts to be stable after 2 rounds. This may be attributed to the fact that the AE trained on the clients provide some pre-trained layers, which capture relevant features. However, we can see that increasing the number of rounds does not always lead to better performance. This is because the model can overfit with large R.

4) Impact of the unlabeled data available on the clients: To evaluate the impact of the quantity of unlabeled data on the model performance, we further fix the amount of labeled data and change only the quantity of unlabeled data distributed to the clients. To do so, we trained our model using different ratios of unlabeled samples R_u:

$$R_u = \frac{nb\_rows(D^u)}{nb\_rows(D^l)} \tag{4}$$

Where $nb\_rows(D^l)$ (resp. $nb\_rows(D^u)$) represents the amount labeled (resp. unlabeled) data.

It can be seen from Figure 5 that increasing the size of unlabeled data improves the performance of the whole model.

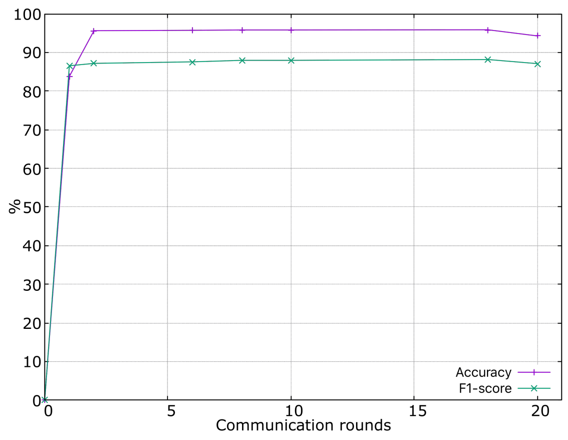


Fig. 4: Classification performance under different communication rounds

These results are attributed to the fact that accessing more (diverse) unlabeled data provides informative characteristics to find a more discriminatory latent space and, in turn, our model benefits from these data and boosts its performance to classify unseen observations. Consequently, our model can fit very well the Industry 4.0 context where unlabeled data is often abundant and easily available.

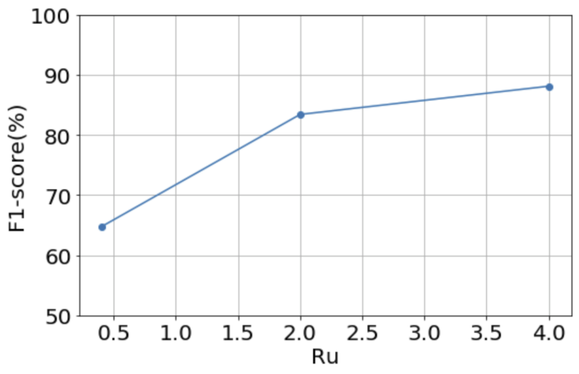


Fig. 5: F1-score with various R_u.

D. Performance against other models

In order to further validate the effectiveness of our model, we compare its performance against the Non-FL semi-supervised model, different supervised models as well as some state-of-the-art models.

1) Comparison with Non-FL model: Evaluation results of our proposed model in comparison to the non-FL model are presented in Table VII. With the non-FL model, the clients/devices need to send their data to a central server in order to train the AE and the supervised models. The below formulas give the training process of our model and the non-FL model where $R = 18$ (total communication rounds), $E_c = 20$ (client AE epochs) and $E_s = 100$ (server supervised learning epochs). T_{FL} (resp. T_{nFL}) is the training process of our FL model (resp. the equivalent non-FL version of our model).

$$T_{FL} = R \times (E_c + E_s) \tag{5}$$

$$T_{nFL} = E_c + E_s \quad (6)$$

In comparison with the non-FL model, our model has the best results in terms of *accuracy* and *F1-score*. This is because the communication rounds help our model to improve the AE model and in turn extract more relevant features that have been used during the supervised model training.

TABLE VII: Performance of the proposed model against the Non-FL model.

Metric	Non-FL model	**Proposed model**
Accuracy	95.4% $\pm$0.34	**95.84% $\pm$0.02**
F1-score	86.7% $\pm$ 0.44	**88.14% $\pm$0.16**

Further, since the communication overhead is a critical factor in the IIoT environment, we compare our model against the Non-FL model in a such factor. Specifically, the communication overhead of both models are calculated as follows (C_{FL} – resp. C_{nFL} – is the communication overhead of our model – resp. the non-FL version of our model):

$$C_{FL} = n \times R \times (2 \times size(AE) + size(H)) \quad (7)$$

$$C_{nFL} = size(D^u) \quad (8)$$

where $size(D^u)$ is the size of the private unlabeled data exchanged between the server and the clients in the Non-FL training, $size(AE)$ is the size of the model located on the client and exchanged between the client and the server in each communication round, n is the number of the selected clients, and R is the total communication rounds, $size(H)$ is the size of the supervised model distributed by the server to client for traffic classification. It can be seen from Figure 6 that centralized learning is always expensive and thanks to the FL, the communication overhead is reduced by **50%**. This is because the clients need to upload their unlabeled data to constitute a central database (Equation 8) while with our model, the clients and cloud server exchange only the models' parameters. In addition, since the AE compresses the client local data this can reduce the size of the parameters communicated with the FL server.

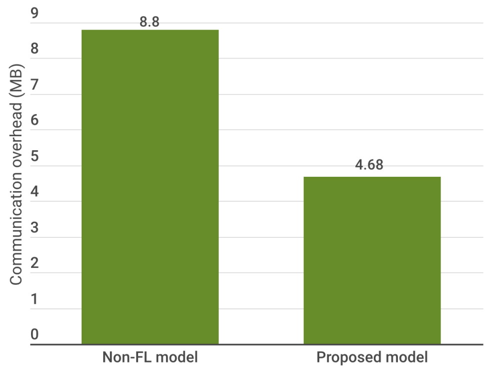


Fig. 6: Comparison in terms of communication overhead.

2) Comparison with supervised models: In order to evaluate the effectiveness of the proposed model, we compare its performance to several supervised ML methods. It can be seen from Table VIII that our model has a good performance in detecting benign network traffic and the different attacks. This is thanks to the use of deep architecture, which covers benign and attacks patterns. Specifically, the F1-score of benign of our model is **6%**, **20%**, **21%**, **21%**, **27%** better than AdaBoost, LightGBM, EXTree, RF, and DT, respectively. This demonstrates that our model is more practical in the Industry 4.0 context as it will trigger fewer false alarms. On the other hand, it achieves a greater F1-score for the classification of the attack, except the *DoS* and especially with *NMRI* attack and this is because the observations amount of this attack is relatively low in the labeled set. From these experiments, we can conclude that our model is better at covering all the attacks at once. Also, using unlabeled data, which is often abundant and easily available improves the classification performance.

3) Performance against state-of-the-art models: To validate the effectiveness of the proposed model, we also compare its performance against some state-of-the-art schemes using the gas pipeline dataset. The experimental results of these schemes are presented in Table IX and they include a simple model with fully labeled data [22], DL-based ensemble model using fully labeled data [23], a semi-supervised model without FL [24], and a supervised ensemble FL scheme [11]. Note that we have selected these works because of their variety. Anton et al. [22] used SVM for intrusion detection. Huda et al. [23] proposed an ensemble Deep Belief Network (DBN) model for attack classification. In particular, different structures of DBN are combined to construct an ensemble of DBNs and the final classification is decided based on a majority voting scheme. Also, Chang et al. [24] proposed an ensemble semi-supervised model using the k-means and convolutional autoencoder (CAE) methods. Using this ensemble, the test data is predicted as normal only if the predicted outputs of k-means and CAE methods are normal. Recently, Li et al. [11] proposed a novel FL model, called `DeepFed`. `DeepFed` is an ensemble model that trains CNN and GRU in a federated way to detect the attacks.

We can observe from Table IX that the intrusion detection model based on FL including our model and `DeepFed` scheme [11] incur the best results. This is due to the communication round that improves the performance of traffic classification to some extent. More specifically, although our model uses a few amounts (only 25%) of labeled data during the training task, it still achieves a competitive accuracy as compared with `DeepFed`. Moreover, with our approach, the clients only train the AE model, which in turn is a less complex model as compared to the CNN-GRU models used with the `DeepFed` scheme. Note that a fair comparison for our model will be to compare it with only semi-supervised schemes. Nonetheless, we include some fully supervised schemes in our comparison for reference.

4) Experiments on the second dataset: We have also tested our model against its non-FL version on the second dataset, which is also an open dataset and has been released by Mississippi State University's lab in 2014 [20]. The traffic

TABLE VIII: F1-score comparison of our model vs. supervised models for the identification of normal vs. attack traffic.

Model	**Benign**	**NMRI**	**CMRI**	**MSCI**	**MPCI**	**MFCI**	**DoS**	**Rec**
DT	0.70	0.32	0.94	0.94	0.97	0.98	**0.98**	**1**
RF	0.76	0.38	**0.96**	0.95	0.97	0.98	**0.98**	**1**
EXTree	0.76	0.49	0.92	0.95	0.95	0.97	0.87	**1**
AdaBoost	0.91	**0.89**	0.92	0.93	0.95	0.97	0.96	**1**
LightGBM	0.77	0.50	**0.96**	0.96	0.97	**0.99**	0.95	**1**
Proposed model	**0.97**	0.20	**0.96**	**0.97**	**0.98**	**0.99**	0.97	**1**

TABLE IX: Overall performance analysis of the proposed model with existing schemes.

Type	**Ref.**	**FL**	**Accuracy**	**Precision**	**Recall**
Supervised	[22]		92.5	78.2	93.6
	[23]		95.6	85.36	85.53
	[11]	✓	99.20	98.85	97.45
Semi-supervised	[24]		95.53	95.43	83.52
	Our model	✓	95.84	97.89	87.15

in this dataset corresponds to the water storage tank control system and consists of 23 features. The label contains eight possible values, benign and seven different types of attacks, same as the gas pipeline dataset.

From the simulation results presented in Table X and Figure 7, we can see that our model performs better than the non-FL model in terms of accuracy, F1-score as well as communication overhead.

TABLE X: Performance of the proposed model against the Non-FL model.

Metric	Non-FL model	**Proposed model**
Accuracy	90.41 $\pm$0.24	**90.73 $\pm$0.08**
F1-score	85.99 $\pm$0.34	**86.41 $\pm$0.09**

Specifically, the improvement in terms of communication overhead becomes more significant with this dataset (reduction by almost 75%), as it contains more traffic than the gas pipeline system and hence sending row data to the central entity becomes more expansive. This shows that our model is suitable for a real IIoT scenario because industrial machines and robots can generate tremendous amounts of traffic.

IV. Discussion

In this study, a semi-supervised FL model is proposed for attack and intrusion detection for the Industry 4.0 environment that makes use of both unlabeled and labeled data. We have analyzed the impact of the different parameters on the performance of the proposed model. First, the evaluation demonstrates that this model performs well even with a limited amount of labels. It automatically provides feature extraction without human intervention and avoids time-wasting for labeling data as maximum as possible. Second, communication overhead, storage requirements have been reduced thanks to the use of FL. Also, we have demonstrated that the local epochs and the clients' selection process play a critical role in communication overhead improvement. Third, using joint announcement protocol addresses the problem of communication

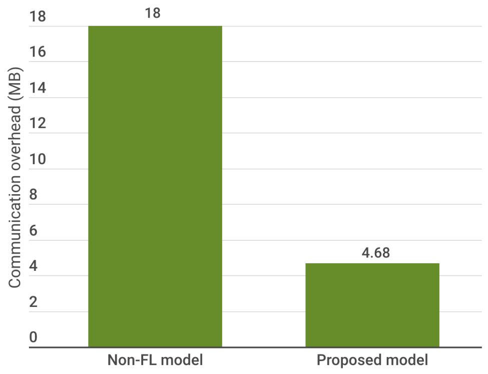


Fig. 7: Comparison in terms of communication overhead.

overhead and the failure of some clients as well as alleviates the out-of-sync issue. Therefore, the communication overhead can be reduced by reducing the number of the selected clients n, reducing the frequency model update using a large number of local client epochs E_c. In addition, taking the advantage of FL our model solves the dilemma of data sharing.

Last but not least, to show the features of our model, we have compared our model against its Non-FL setting, some state-of-the-art models including, simple, ensemble, semi-supervised, and FL models. Also, we applied our model on a second dataset and evaluated its performance in terms of accuracy, F1-score, and communication overhead. The presented results show that our model achieves higher classification with less communication overhead as well as without privacy concerns.

Limitations of proposed scheme

Although our model uses the joint-announcement protocol, the random client selection can increase the training time and communication cost due to the clients who become stragglers. To handle these issues, more intelligent client selection algorithms are needed. One idea could be to use reinforcement learning, which can be a promising solution, to learn client selection based on the learning performance. Also, as our model still need labels in future research, we can investigate the fully unsupervised FL model by looking at the AE reconstruction error. Finally, to ensure a secure model transmission, we can use some mechanisms such as differential privacy [25] or blockchain [26].

V. Conclusion

In this paper, a federated semi-supervised learning model has been proposed. This model uses a limited amount of labeled data and a huge amount of unlabeled data without privacy concerns. Also, unlike the classical FL model, with our model, the server is not only used for the model aggregation task, but also for supervised learning. The proposed model has been evaluated in terms of its ability to identify the network traffic and different attacks. Moreover, we have analyzed the performance of our model while varying different factors. Using two real datasets, the experimental results demonstrate that the support of unlabeled data for the training process can enhance the performance of the learned model as well as decrease the communication overhead. Also, our numerical simulations showed that the proposed federated semi-supervised model with only 25% labeled data can achieve competitive results, compared to the state-of-the-art schemes.

References

[1] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, D. Niyato, and H. V. Poor, "Federated Learning for Industrial Internet of Things in future industries," *arXiv preprint arXiv:2105.14659*, 2021.

[2] N. Mishra and S. Pandya, "Internet of Things applications, security challenges, attacks, Intrusion Detection, and future visions: A systematic review," *IEEE Access*, 2021.

[3] W. Wang, M. H. Fida, Z. Lian, Z. Yin, Q.-V. Pham, T. R. Gadekallu, K. Dev, and C. Su, "Secure-enhanced Federated Learning for AI-empowered electric vehicle energy prediction," *IEEE Consumer Electronics Magazine*, 2021.

[4] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 1273–1282.

[5] M. Ge, X. Fu, N. Syed, Z. Baig, G. Teo, and A. Robles-Kelly, "Deep Learning-based Intrusion Detection for IoT networks," in *2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC)*. IEEE, 2019, pp. 256–25 609.

[6] P. Kumar, G. P. Gupta, and R. Tripathi, "A distributed ensemble design based Intrusion Detection System using fog computing to protect the Internet of Things networks," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–18, 2020.

[7] M. Aamir and S. M. A. Zaidi, "Clustering based Semi-Supervised Machine Learning for DDoS attack classification," *Journal of King Saud University-Computer and Information Sciences*, 2019.

[8] K. Hara and K. Shiomoto, "Intrusion Detection System using Semi-Supervised Learning with Adversarial Auto-encoder," in *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2020, pp. 1–8.

[9] V. Mothukuri, P. Khare, R. M. Parizi, S. Pouriyeh, A. Dehghantanha, and G. Srivastava, "Federated Learning-based anomaly detection for IoT security attacks," *IEEE Internet of Things Journal*, 2021.

[10] D. C. Attota, V. Mothukuri, R. M. Parizi, and S. Pouriyeh, "An ensemble multi-view Federated Learning Intrusion Detection for IoT," *IEEE Access*, 2021.

[11] 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 Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5615–5624, 2020.

[12] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, "DÏoT: A federated self-learning anomaly detection system for IoT," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2019, pp. 756–767.

[13] Y. Fan, Y. Li, M. Zhan, H. Cui, and Y. Zhang, "IoTDefender: A Federated transfer Learning intrusion detection framework for 5G IoT," in *2020 IEEE 14th International Conference on Big Data Science and Engineering (BigDataSE)*. IEEE, 2020, pp. 88–95.

[14] S. I. Popoola, R. Ande, B. Adebisi, G. Gui, M. Hammoudeh, and O. Jogunola, "Federated deep Learning for Zero-Day botnet attack detection in IoT edge devices," *IEEE Internet of Things Journal*, 2021.

[15] M. A. Ferrag, O. Friha, L. Maglaras, H. Janicke, and L. Shu, "Federated Deep Learning for cyber security in the Internet of Things: Concepts, applications, and experimental analysis," *IEEE Access*, 2021.

[16] M. Sarhan, S. Layeghy, N. Moustafa, and M. Portmann, "A cyber threat intelligence sharing scheme based on Federated Learning for Network Intrusion Detection," *arXiv preprint arXiv:2111.02791*, 2021.

[17] R. Taheri, M. Shojafar, M. Alazab, and R. Tafazolli, "FED-IIoT: A robust federated malware detection architecture in industrial IoT," *IEEE Transactions on Industrial Informatics*, 2020.

[18] Y. Liu, J. James, J. Kang, D. Niyato, and S. Zhang, "Privacy-preserving traffic flow prediction: A Federated Learning approach," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7751–7763, 2020.

[19] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew, "Deep Learning for visual understanding: A review," *Neurocomputing*, vol. 187, pp. 27–48, 2016.

[20] T. Morris and W. Gao, "Industrial control system traffic data sets for intrusion detection research," in *International Conference on Critical Infrastructure Protection*. Springer, 2014, pp. 65–78.

[21] S. Agrawal, S. Sarkar, O. Aouedi, G. Yenduri, K. Piamrat, S. Bhattacharya, P. K. R. Maddikunta, and T. R. Gadekallu, "Federated Learning for Intrusion Detection System: Concepts, challenges and future directions," *arXiv preprint arXiv:2106.09527*, 2021.

[22] S. D. D. Anton, S. Sinha, and H. D. Schotten, "Anomaly-based Intrusion Detection in industrial data with SVM and Random Forests," in *2019 International conference on software, telecommunications and computer networks (SoftCOM)*. IEEE, 2019, pp. 1–6.

[23] S. Huda, J. Yearwood, M. M. Hassan, and A. Almogren, "Securing the operations in SCADA-IoT platform based industrial control system using ensemble of Deep Belief

Networks," *Applied soft computing*, vol. 71, pp. 66–77, 2018.

[24] C.-P. Chang, W.-C. Hsu, and I.-E. Liao, "Anomaly Detection for industrial control systems using k-means and Convolutional AutoEncoder," in *2019 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. IEEE, 2019, pp. 1–6.

[25] Z. Lian, W. Wang, and C. Su, "COFEL: Communication-efficient and optimized Federated Learning with Local Differential Privacy," in *ICC 2021-IEEE International Conference on Communications*. IEEE, 2021, pp. 1–6.

[26] W. Wang, H. Xu, M. Alazab, T. R. Gadekallu, Z. Han, and C. Su, "Blockchain-based reliable and efficient certificateless signature for IIoT devices," *IEEE Transactions on Industrial Informatics*, 2021.