2023 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech) | 979-8-3503-0460-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/DASC/PICOM/CBDCOM/CY59711.2023.10361504

FL-SERENADE: Federated Learning for SEmi-supeRvisEd Network Anomaly DEtection. A case study

Veronika Bekbulatova
TU Wien
Vienna, Austria
veronika.bekbulatova@gmail.com

Andrea Morichetta
Schahram Dustdar
Distributed Systems Group, TU Wien
Vienna, Austria
lastname@dsg.tuwien.ac.at

***Abstract*—The use of connected devices in the industry represents a necessity and, at the same time, a challenge. Building a network of interconnected industry assets can improve performance and scale but can lead to dangerous security risks and attacks. However, the amount of data shared, and the widespread distribution of devices make the protection of industrial resources cumbersome. One problem is to know the type of information flowing and check for anomalies, making the job of anomaly-based Intrusion Detection Systems (IDSs) arduous. In this direction, we explore a semi-supervised approach, "Deep-SAD," to overcome the partial knowledge of the data. Due to the size of the data, and the need for privacy measures, we combine this model with a federated learning (FL) framework "Flower," distributing the learning phase through five industrial areas. We evaluate our implementation over the WUSTL-IIoT-2021 dataset, a testbed simulation of an actual plant where threats have been injected. This work presents and evaluates a framework for semi-supervised anomaly detection, starting with feature engineering. The results reveal that the difference in the performance of the federated and centralized settings is minimal, denoting the promising application of the federated approach. Combined with the security and privacy-preserving characteristics of FL, this demonstrates the value of the federated approach to the semi-supervised anomaly-based IDS in the IIoT networks.**

***Index Terms*—Internet of Things (IoT); Security and Privacy; DoS attacks; Federated Learning**

I. Introduction

The ongoing adoption of information technology in the industrial infrastructure is the primary driver of the emerging fourth industrial revolution, also known as Industry 4.0. It aims to radically change industrial sites by interconnecting smart devices, workers, suppliers, and customers to create intelligent systems capable of advanced analytics and decision-making. Industrial IoT is an essential component of Industry 4.0. In a broad sense, it refers to enhancing industrial systems with "smart" devices such as sensors, actuators, and RFID tags. However, hardware and software used in IoT devices are very heterogeneous, and the current technology needs to improve in consistent security standardization and risk assessment norms [1], [2]. At the same time, industrial networks usually leverage legacy systems designed with different requirements, as these systems were traditionally detached from the Internet, the so-called operation technology (OT) sector. As a result, attacks against industrial systems are on the rise due to the larger, more complex attack surface [3]. As the Kaspersky ICS CERT report from 2021 [4], out of all industrial computers that used Kaspersky tools, 39.6% received attacks during the year. Many industrial sites belong to critical infrastructure, and an interruption to their service might result in enormous costs and even lead to catastrophic events. For example, cyber-attacks on the Ukrainian power grid in 2015 resulted in a massive power outage that continued for multiple hours [5]. Therefore, Denial of Service attacks (DoSs) represent a real threat. However, tackling them in such a scenario is cumbersome [6], [7], [8], [9], [10], [11]. In this context, anomaly-based Intrusion Detection Systems (IDSs) can model the general "regular" traffic and detect unusual activity. Modern IDSs usually require developing and training machine learning (ML) models, demanding large datasets and extensive computing resources. Hence, the deployment of ML applications typically occurs in a cloud environment, often outsourced to third-party vendors. This way, the data flows through the network to the central cloud location for ML training. However, this approach comes with shortcomings in the case of industrial IDS [12]. First, sharing the network traffic with a third-party cloud can breach privacy regulations, mainly if the provider resides in a different country. Furthermore, connecting all end nodes to the Internet and transferring confidential and sensitive data creates additional vulnerabilities. Finally, uploading the massive industrial network data to a distant cloud location results in high bandwidth costs and latency issues [13] in critical industry scenarios with real-time constraints, such as power grids and nuclear plants.

Federated Learning holds the promise to solve the main drawbacks mentioned above. Models train locally on each end node, and only the model parameters are shared for further cloud aggregation. The cloud server builds a more accurate global model by aggregating the received weights and sending the outcome back to the edge devices for anomaly detection in the edge network traffic. After the introduction by McMahan

979-8-3503-0460-2/23/$31.00 ©2023 IEEE

et al. in [14] in 2016, FL became a popular topic in academia and industry. However, there are considerably fewer studies regarding FL application to anomaly-based IDS in IIoT. In addition, considering the humongous dimension of industrial network traffic, more than supervised learning is needed, as labeling all the items is impractical. However, alternatives like semi-supervised approaches in FL are understudied. In reality, dealing with partially labeled sets offers performance advantages and makes the process more manageable. Furthermore, public open-source FL frameworks, like Flower [15], are overlooked. Instead, they can foster the research in a production-ready scenario. Flower can handle large-scale heterogeneous devices and is agnostic regarding hardware, ML frameworks, programming languages, and communication protocols.

Therefore, this work aims to collect the pinpointed gaps, inspecting a federated semi-supervised approach for IDS model training in IIoT, specifically for DoS attacks. We offer a methodology for semi-supervised deep learning using a novel model called Deep-SAD, where we define the main challenges and solution. Furthermore, we test the framework in a federated learning scenario. Furthermore, we evaluate it through a case study with a rich security dataset, showing the promising direction of this approach and set the foundation for production IDSs. Section II briefly overviews the theoretical background for IIoT security and semi-supervised anomaly detection for IDS and FL. Firstly Section III is dedicated to the methodology, where we present the semi-supervised anomaly detection and FL frameworks, the dataset, and the performance metrics. Section IV discusses the frameworks' optimization, focusing on configurations and experiments on the case study. Furthermore, it provides the experimental results and compares the performance of centralized and federated setups. Section V concludes the paper, discussing the limitations and possible future research directions.

II. Background and Motivation

We propose a federated semi-supervised anomaly-based approach to IDS for IIoT security. complexity, lack of standardization, massive data, and privacy regulations

a) Overview: Security in Industrial IoT: IIoT is the interconnection of internet-connected devices and industrial machines, who collaborate [16, p.1]. However, the exposure to the network lead to many security issues in IIoT, with more frequent external cyber-attacks [17] that might expose sensitive data and can even directly impact human health and safety [5]. There needs to be more than conventional IT cybersecurity measures to address such a scenario characterized by largely heterogeneous devices. In the context of external attacks, an Intrusion Detection System (IDS) can be used to check IIoT systems for suspicious activity and possible intrusions. However, this scenario has some challenges that typically undermine the IDS efficacy.

- **Challenge 1**. IIoT systems are complex and lack standardization, making building rules to monitor and keep track of the highly dynamic threat landscape difficult. [2]. *Possible Solution:* anomaly-based IDS powered by ML [18].
- **Challenge 2**. IIoT systems produce a massive amount of heterogeneous data, which makes it barely feasible to label huge traffic datasets. *Possible Solution:* Semi-Supervised Learning(SSL) [18].
- **Challenge 3.** Having IDS models in the cloud requires extensive data transfer over the Internet from the IIoT end nodes to a remote central location. Other than latency issues, this approach creates additional cyber-attack vectors and potentially violates privacy regulations such as GDPR.

To overcome the challenges above, we propose a federated semi-supervised anomaly-based approach to IDS for IIoT security. We provide theoretical justifications for each constituent method, including anomaly-based IDS, SSL, and Federated Learning (FL) [18], [19].

b) Anomaly-based IDS: There exist various categories of IDSs. Given the challenges of the IIoT scenario, it is essential to make the most suitable decisions while designing one. Since IIoT devices are often resource-constrained, we focus on network-based IDSs that, conversely to host-based, reside on network devices, capturing the traffic flowing through them [20]. Furthermore, IDSs can be *signature-based* and *anomaly-based*, depending on how they tackle intrusions. Signature-based IDSs compare the observed activity to known attack patterns. When matched, they produce an alarm. Thus, this category requires constant maintenance and updates due to the rapidly evolving threat landscape. Unknown attacks, like zero-day vulnerabilities, or attacks to new categories of devices are out of sight for the standard signature-based IDS. Anomaly-based IDSs, on the other hand, model their inspector over regular activity; they raise the alarm whenever something deviating occurs [21]. Therefore, network- and anomaly-based IDSs represent the most suitable solution for IIoT, undertaking the heterogeneity of devices and behaviors, and addressing **Challenge 1**.

c) Semi-Supervised Deep Learning for Anomaly Detection: Compared to many other ML techniques, DL does not require extensive feature engineering by human experts and can process raw data input. As traffic volumes are universally increasing, including in IIoT systems, DL is a promising approach to anomaly-based IDS, as the traffic data directly inputs in the model training [22]. Based on the training data, DL, like other ML techniques, can be divided into *supervised, unsupervised, and semi-supervised learning*. Semi-Supervised Learning(SSL) falls between the first two categories as it can operate on mixed types of training data. However, as Engelen et al. [23] state, semi-supervised learning needs unlabeled samples to contain new information that can enrich the model knowledge. In IIoT networks, a large volume of unlabeled standard traffic is typical, alongside rare anomalies. In this scenario, it is possible to obtain labeled anomalies, e.g., through red teaming exercises or domain knowledge by security experts. Thus, unlabeled data is essential as it reveals new information on the expected system behavior, aligning with Engelen et al.'s perspective. Applying semi-supervised learning to the anomaly-

based IDS tackles **Challenge 2**, by providing autonomous and sophisticated models to identify security breaches.

d) Federated Learning: Federated Learning (FL) is a privacy-preserving approach proposed by Google researchers in 2016 that enables Machine Learning (ML) with distributed data sources [14]. Since then, FL has received considerable attention, being applied to various applications [24], [25], [26]. A relevant characteristic of FL is enabling collaboration between various entities, such as end devices [27] and independent organizations [28], without exposing their sensitive or proprietary data. This characteristic makes FL suitable for the edge computing paradigm. It has advantages for IoT systems [29], [30], [31], and IIoT in particular, and can help tackle **Challenge 3** [19]. Semi-supervised anomaly detection with FL includes various label distribution scenarios, such as Label-centralized Federated SSL and Label-distributed Federated SSL [32]. This work focuses on the Label-distributed Federated SSL, where each client has approximately the same amount of labeled and unlabeled data, fitting the Labels-at-Client Scenario [33].

III. Methodology

A. WUSTL-IIoT-2021 Dataset

Currently, public datasets that come from actual industrial environments are scarce. According to [5], [34], industrial companies have multiple reasons, such as (1) user privacy regulations; (2) confidentiality policies related to intellectual property and adopted technological solutions; (3) concerns about exposing internal security and safety architectures to potential intruders. Furthermore, real-world datasets might not be suitable for security research due to insufficient attack scenarios. The alternative is to use simulations building testbeds [5]. One of such physical testbeds for industrial cybersecurity research, set up by Teixeira et al. in [35] and further extended by Zolanvari et al. in [34]. It is our choice for the evaluation. It comprises a water tank, sensors, actuators, and the underlying Modbus network. The idea is to simulate and resemble an industrial plant for a water supervision system. The traffic was captured for 52 hours, during which the researchers carried out four types of cyber-attacks: backdoor, command injection, reconnaissance, and DoS attacks, the latter being the predominant one in the dataset. The resulting dataset was cleaned from missing and corrupted values and made available as *WUSTL-IIoT 2021 Dataset*[36]. This work references the dataset as the *IIoT dataset*. The original IIoT dataset consists of 1 194 464 samples, and 49 features. [1]

B. Handle Categorical features

The dataset contains three categorical features, stored as integers: *Source Port, Destination Port, and Protocol*, nominal and high-cardinal. Therefore they cannot be easily encoded with methods such as one-hot encoding, which would increase the input cardinality. Moreover, they do not capture the relationship between categories. The authors of the IIoT dataset train various ML models in [34], but they do not mention any pre-processing of the categorical features. Therefore, we suggest the IIoT dataset needs a better approach to handle its categorical data. In this work, we present two possible approaches to this issue. The first solution, defined as *Framework 1*, includes removing the categorical features from the dataset to rely only on the numerical data. The second solution, *Framework 2*, handles the categorical features with the help of *Entity Embeddings*.

Framework 1. Keeping only the numerical features and removing the categorical ones simplifies the application of ML algorithms. However, Zolanvari et al. [36] examined the importance of each feature by removing one feature per training and by measuring the change in performance. *Source Port* and *Destination Port* are among the five most important features. We use a different model in the evaluation but might still skip some relevant information.

Framework 2. An embedding is the compression of multi-dimensional data into a lower-dimensional space. *Entity Embedding* is a term coined by Guo and Berkhahn [37] that defines a representation of categorical features as lower-dimensional vectors in the Euclidean space; the distance between the vectors captures meaningful relations between the categories. Each categorical feature is one-hot encoded and passed through a neural layer – called the embedding layer. The output of all embedding layers and the numerical features are concatenated and used as the input to the main classification model. The neural layers learn the mappings of the categorical data to the numerical vectors during the model's training process.

We implement the embedding layers developing on Seth et al. [38]. In our case, the embedding layer is not preceded by a dropout layer but rather concatenated with the numerical features and connected to the neural network. A critical embedding parameter is the resulting numerical vector dimension. In the experiment, we follow the so-called "rule of thumb [2]":

$$dims = min(600, 1.6 \cdot n\_cat^{0.56}) \quad (1)$$

where *dims* is the resulting vector dimensionality and *n_cat* is the cardinality of the categorical feature. We present the resulting dimensions of the categorical features in Table I. This approach is more complex than Framework 1, impacting the training time.

Feature	Cardinality	Embedding Dimensions
Source Port	51047	600
Destination Port	33	11
Protocol	7	5

TABLE I: Embedding dimensions of the categorical features

C. Anomaly Detection Framework

In the search process for a suitable AD framework, we inspect methodologies able to work with partial knowledge of the data, i.e., capable of semi-supervised learning. Furthermore, the model must be capable of processing non-image datasets.

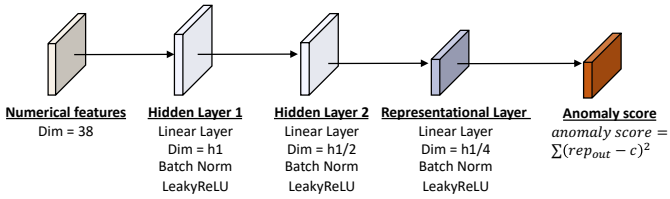
[1]As advised in [36], the following features are removed from the dataset: *StartTime, LastTime, SrcAddr, DstAddr, sIpId, dIpId*.

[2]https://github.com/fastai/fastai/blob/master/fastai/tabular/model.py

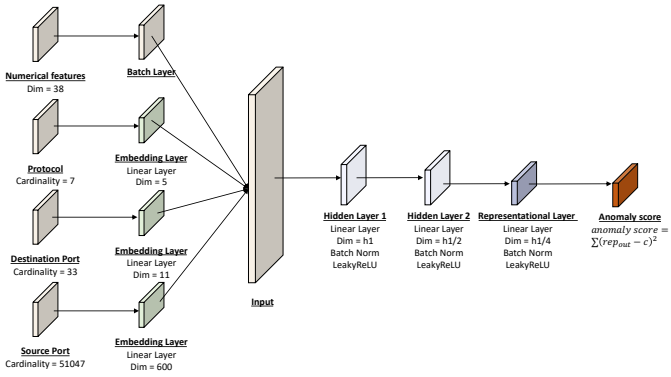
Many available frameworks in AD research mainly process image data [39]. Given these prerequisites, we choose "Deep Semi-Supervised Anomaly Detection" (Deep-SAD). [3] Deep-SAD compresses high-dimensional input to a low-dimensional vector space by passing it through the neural layers. The vector output of the last layer is the latent representation of the input data point [23]. The model then maps similar data points close within the latent space. The idea is to map all regular data points on a hypersphere, as close as possible to a fixed point c, representing the central point of all regular samples. At the same time, the anomalous data is mapped as further away from the center of the hypersphere so that it can be discriminated based on the distance to c. The center c is initialized by passing the training data through the initially untrained network, finding the mean value for the output vector. The initial network is initialized randomly or via pre-training with an autoencoder. The semi-supervised learning in DeepSAD assumes that anomalies are infrequent events and that most unlabeled data lie in the regular class. Therefore, the model is trained to map unlabeled samples closer to the center c. Once trained, the model outputs anomaly scores for each input sample. Therefore, it is necessary to fix a specific threshold for the anomaly score to separate instances into two classes (regular and anomalous) for binary classification.

a) Deep-SAD parameters: Deep-SAD changes the labels of regular labeled data to 1, anomalous labeled data to -1, and unlabeled data to 0. The framework enables supervised, unsupervised, and semi-supervised learning by three parameters. The first two, namely *ratio of labeled regular samples* and *ratio of labeled anomalous samples*, regulate the percentage of labeled points for regular and anomalous points over the whole set. The third, *ratio of unlabeled anomalies in the unlabeled data (pollution ratio)*, controls how much of the unlabeled subset should contain known anomalous points. Furthermore, a fourth parameter η defines whether labeled or unlabeled samples are more emphasized during the training. $\eta > 1$ gives more weight to the labeled data, $\eta < 1$ emphasizes the unlabeled data and $\eta = 1$ treats both types of data as equally important.

1) Neural Architecture: The authors of Deep SAD [39] utilized different neural architectures for each test dataset. As a template, we use the structure for the ODDS datasets[4], standard benchmarks for AD research, and tabular, like our IIoT dataset. Similarly to [39], we use a neural network with one input layer, two hidden layers(*h1,h2*), and one representational layer(*rep*). The input layer consists of the same number of neurons as the number of input features; for other layers, the following rules are used: $h2 = \frac{h1}{2}$, $rep = \frac{h1}{4}$. Thus, only *h1* requires a fixed value, while other neural layers are derived from it. To find a suitable number of neurons in (*h1*) we define a set of possible values for the hyperparameter optimization; we discuss this in the case study evaluation IV. As we have two frameworks for categorical data handling, their neural architectures are different, i.e., framework 2 has additional embedding layers. The neural architectures of the frameworks are demonstrated in Figure 1.



(a) Framework 1



(b) Framework 2

Fig. 1: Neural architectures of Framework 1 and Framework 2

2) Federated Learning Framework: To evaluate the federated setting, we select the open-source framework Flower [15]. Compared to other platforms, like OpenFL [5] – which is more security oriented – Flower exhibits better maturity and documentation. It is worth noticing, though, that OpenFL and Flower will collaborate, so some characteristics of the former will be included in Flower. Flower interfaces can be extended and adjusted for a specific production or research objective. Furthermore, Flower is agnostic regarding programming languages, ML frameworks(TensorFlow, PyTorch, and others), communication approaches, and underlying hardware. This characteristic is advantageous in the case of heterogeneous clients. These properties of Flower make it a suitable framework for IIoT applications. Another advantage mentioned is the capability to move ML models from a centralized setting to the federated one.

a) Federated Deep-SAD with Flower: The transition of an existing ML model to the FL setting is facilitated by the high-level interface *FlowerClient*. The server side of FL is supported by high-level interfaces *FlowerServer* and *Strategy*. This FlowerClient class comes with functions such as *fit*, *set_parameters*, and *evaluate*. We override the functions to invoke the respective Deep-SAD training and testing routines, thereby embedding the model in FlowerClient. No further changes to the underlying Deep-SAD architecture are needed. A separate FlowerClient instance is created and loaded in memory for each federated client, making it unsuitable for

[3]https://github.com/lukasruff/Deep-SAD-PyTorch

[4]http://odds.cs.stonybrook.edu/

[5]https://github.com/securefederatedai/openfl

large-scale experiments. Therefore, a new workflow, enabled by the Flower's Virtual Client Engine(VCE) tool, solves this issue. [6]. VCE instantiates and keeps an actual client object only during client tasks, such as training or testing. After the task completion, VCE erases the client object, resulting in higher memory efficiency. Still, due to the Deep-SAD internal structure, client instances must be preserved in the memory so that the parameters, such as the hypersphere center c, can be exchanged between training and testing. This limitation could be bypassed by adjusting the implementation of Deep-SAD; however, it is out of the scope of this paper. The FlowerServer class leverages all the necessary server-side architecture. Strategy is one of the major components; it implements the client selection, model aggregation, and model evaluation routines. Flower provides built-in algorithms but also allows for modifications and extensions. In this work, we leverage the default architectures.

D. Performance Metrics

We use performance metrics in multiple experimental phases, including selecting hyperparameters and evaluating the centralized and federated models. We measure performance using the AUC-ROC metric, which calculates the area under the ROC curve, and the AUC-PR metric, which calculates the area under the Precision-Recall curve. AUC-PR is crucial for highly skewed datasets, providing additional insights into False Positives. F-score is not used in performance evaluation, as it is sensitive to the anomalous class ratio and requires a fixed threshold for the anomaly score. Instead, we use AUC-ROC and AUC-PR to evaluate performance. [7]

IV. Case Study

Here we conduct a series of experiments both in centralized and federated settings for our IIoT case study. All experiments are conducted on a desktop computer with Python 3.7, Windows 10 Pro operating system, Intel©Core™i7-4790K CPU running at 4.0 GHz, 8 GB of RAM, and one NVIDIA GeForce GTX 970.

A. Dataset Pre-processing

The dataset contains four different attack types. However, this work focuses on the binary classification of anomalies. Therefore, we keep only regular and DoS samples, as the DoS attacks produced the most anomalous traffic. We have more than 1 million samples, with the malicious samples (DoS) equal to 6.6% of the whole dataset. In all subsequent experiments, we utilize 60% of the dataset for training and 40% for testing. We apply the Z-score standardization to each numerical feature and scale the range to [0,1]. Categorical features are untouched. We conduct the experiments in the *semi-supervised setting* assuming that most of the data in real-world scenarios are unlabeled and that most unlabeled data belongs to the regular class. Still, a small fraction of anomalies can be in the unlabeled data. The pollution rate in the training dataset is therefore set to 5%. That also means that 95% of the unlabeled data in training belongs to the regular class. Next, we assume that labeling the regular class would not be prioritized, so there are no labeled regular samples in the training phase. However, obtaining a small training set of labeled anomalies is essential to increase anomaly detection; therefore, 1% of the overall training dataset has labeled anomalies. Moreover, we set $\eta = 1$ so that both labeled and labeled are treated as equally crucial by Deep SAD. The semi-supervised setting is summarized in Table II.

Parameter	Value
η	1
Ratio of normal labeled samples	0% (of the total training dataset)
Ratio of anomalous labeled samples	1%(of the total training dataset)
Pollution rate	5% (of the unlabeled training subset)
Ratio of normal unlabeled samples	95% (of the unlabeled training subset)

TABLE II: Experimental semi-supervised setting

B. Hyperparameter Tuning and Framework selection

The fixed DL parameters used in the experiments are Weight decay: 10^{-6}, Adam optimizer, 50 epochs training, and a learning rate of 25. We use the hyperparameter optimization framework Tune, offered by the distributed computing framework Ray [41]. We tune the hyperparameters for Framework 1 and Framework 2 before comparing their performances. The selected hyperparameters are: Learning Rate: $[0.0001, 0.001, 0.01]$, Batch size: $[64, 128, 256]$, and number of neurons in the first hidden layer: $[64, 128, 256, 512]$. There are 36 possible configurations, Tune trials, where a trial is a single experiment with a fixed set of hyperparameters. We limit the number of trials to 18, which comprises 50% of all possibilities. However, instead of choosing the 18 configurations randomly, we employ Tune's built-in optimization libraries, such as *Search Algorithms* and *Trial Schedulers*. The search algorithm optimizes the parameter selection for each trial by prioritizing the most promising parameters in the queue. Here, we use the HyperOpt algorithm[8]. The trial scheduler is responsible for stopping unpromising trials. In this work, we use the ASHA algorithm. [9] For the trials' evaluation, we utilize the AUC-ROC. That means each trial is compared based on its AUC-ROC. As estimated by Tune after 18 trials, the best AUC-ROC for Framework 1 and 2 were delivered by the following hyperparameters: Framework 1: *(Learning rate: 0.001, batch size: 256, number of neurons: 128)*; Framework 2: *(Learning rate: 0.01, batch size: 64, number of neurons: 128)*.

Having the tuned configurations of Framework 1 and Framework 2, we compare the performances of the frameworks. Framework 1 scores better based on AUC-ROC and AUC-PR. Even though some categorical features belong to the five most essential features according to Zolanvari et al. [36], it was still possible to get excellent results. Another obvious advantage of Framework 1 is the training time, which is 13 times faster

[6]https://flower.dev/docs/changelog.html#v0-17-0-2021-09-24

[7]Davis et al. [40] noted that AUC-ROC might be misleading when applied to heavily imbalanced datasets, and we include the AUC-PR metric for such datasets.

[8]https://docs.ray.io/en/latest/tune/api_docs/suggestion.html

[9]https://docs.ray.io/en/latest/tune/api_docs/schedulers.html

than Framework 2. Adding the embedding layers increases the training time drastically, a significant drawback for time-constrained applications. Thus, we elect Framework 1 as the primary model.

C. Pre-training

In [39], the weights of the Deep-SAD neural network were initialized with the help of an autoencoder(AE), which encoder part has the same neural architecture as the original Deep-SAD network. The encoder part of the trained AE initializes the Deep SAD network weights. This approach offers a better way to initialize weights than random ones. In our case, we use random weight initialization when tuning Framework 1 and 2. After selecting Framework 1, in this Section, we analyze the influence of the autoencoder pre-training on the model performance. We utilize an AE with the same neural architecture in the Encoder part as Framework 1. The overall structure of the AE used for Framework 1 is presented in Figure 2. While the model training without pre-training results in the same performance ($PR = 99.3\%$), the configuration with pre-training takes three times longer [10] which significantly differs in processing time. Therefore we select Framework 1 without the pre-training.

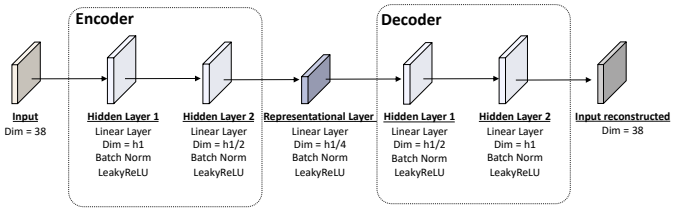


Fig. 2: Neural architecture of the autoencoder used for Framework 1 pre-training.

D. Federated Setup

We use the 0.18 Flower release for the FL experiment, and the training runs as a single-machine simulation. The federated configuration includes 5 Clients and 1 Server. Each client simulates a region in the IIoT scenario and utilizes the exact configuration of Deep SAD as the centralized model. In this case study, we consider the scenario where each client has both labeled and unlabeled data and has a comparably equal subset of the IIoT dataset. Furthermore, each client has the same proportion of unlabeled and labeled data. Table III shows the summary of the clients' datasets.

	Client 1	Client 2	Client 3	Client 4	Client 5
Normal labeled samples in the training set	0	0	0	0	0
Anomalous labeled samples in the training set	1413	1413	1412	1412	1413
Anomalous unlabeled samples in the training set	6996	6994	6992	6991	6996
Normal unlabeled samples in the training set	132940	132900	132849	132840	132938
Total number of samples in the training set	141349	141307	141253	141243	141346
Anomalous samples in the test set	6234	6261	6295	6300	6234
Normal samples in the test set	88628	88600	88566	88561	88626
Total number of samples in the test set	94862	94861	94861	94861	94860
Total number of samples in the full dataset	236211	236168	236114	236104	236206

TABLE III: Summary of IIoT dataset in the experimental semi-supervised setting for each federated client.

[10]Training time: 1405.46s No pre-training, 2661.53s Pretraining; testing time: 12.69s No pre-training, 14.82s Pretraining

The model aggregation algorithm is the built-in FedAvg, a standard baseline algorithm in FL research. Given our scenario, we set to five the following FL parameters in the Flower framework: (i) *min_fit_clients*, the minimum number of clients that need to participate in the training phase; *min_eval_clients*, the minimum number of clients participating in the testing phase; and *min_available_clients*, the minimum number of clients that need to be connected to the server to start a federated round[11]. The evaluation consists of three federated rounds, with the first round performing the following steps: (1) The server randomly initializes the weights of the global model. (2) The server waits until all five clients are available and connected to the server. (3) The server selects all five clients to participate in the training and shares the global model with them. (4) The clients pick and train the model with their local datasets. (5) The clients return their locally trained models to the server. (6) The server aggregates the received models into the updated global model with FedAvg. (7) The server selects all five clients to participate in the testing and shares the updated global model. (8) The clients pick and evaluate the new global model with their local datasets. For the subsequent FL rounds, we repeat steps 2-8. Each client trains the model for 50 epochs per round; thus, in total, 150 epochs.

E. Experimental Results

Here, we evaluate the performance of Framework 1 in centralized and federated settings. The centralized model is first trained with 60% of the whole IIoT data. In the testing phase, the trained model outputs anomaly scores for the remaining 40% of the data. Based on the resulting anomaly scores, we measure AUC-ROC and AUC-PR. For the federated setting, in each round, each client trains the model with 60% of the client's data. Each round has a testing phase, where anomaly scores are produced for the remaining 40% of the data, together with AUC-ROC and AUC-PR. We extract the average AUC-ROC and AUC-PR for all clients after every round to compare the centralized and federated settings. The resulting AUC-ROC for both settings is presented in Table IV, while the resulting AUC-PR is presented in Table V. The results demonstrate that the centralized model produces the best AUC-ROC and AUC-PR. For AUC-ROC, the model even achieves the perfect score of 100%. We consider two possible motivations for that: 1) As mentioned in Section III-D, AUC-ROC can sometimes produce an overly optimistic measure of performance, and it is helpful to measure AUC-PR for additional insights. Indeed, the centralized model achieved a slightly lower but still excellent AUC-PR of 99.93%; 2) We evaluate the models' performance only on DoS samples in the IIoT dataset. The DoS traffic often has distinctive characteristics such as a high rate of packets transmitted per second or only unidirectional communication [42]. We assume the model has quickly learned to distinguish the DoS traffic in the IIoT dataset. Future framework evaluation should include multiple attack types and more sophisticated attack scenarios. However, despite the limitations, this centralized configuration

[11]https://flower.dev/docs/strategies.html

of Deep SAD shows promising results for DoS attack detection in IIoT networks. Even though the centralized model shows the best results, the difference between the two settings is negligible. Moreover, when compared separately with each client, 4 out of 5 clients achieved the same AUC-ROC after three federated rounds. Also, after three federated rounds, 2 out of 5 clients achieved better AUC-PR than the centralized model.

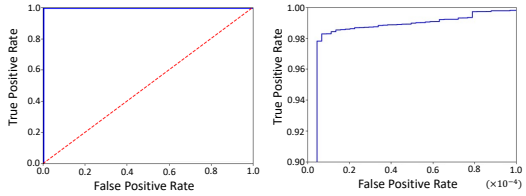
	Round 1	Round 2	Round 3
Client 1	99.98%	100.00%	100.00%
Client 2	99.96%	99.92%	100.00%
Client 3	99.98%	99.99%	100.00%
Client 4	99.65%	99.87%	99.99%
Client 5	99.97%	99.99%	100.00%
Average	99.908%	99.954%	99.998%
Centralized	100%		

TABLE IV: Comparison of AUC-ROC in centralized and federated settings.

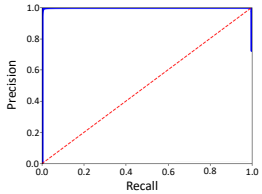
	Round 1	Round 2	Round 3
Client 1	99.74%	99.94%	99.96%
Client 2	99.43%	99.73%	99.91%
Client 3	99.60%	99.76%	99.83%
Client 4	99.49%	99.83%	99.96%
Client 5	99.61%	99.65%	99.71%
Average	99.574%	99.782%	99.874%
Centralized	99.93%		

TABLE V: Comparison of AUC-PR in centralized and federated settings.

The plots of ROC and PR curves measured in the centralized setting are demonstrated in Figure 3. The plots of ROC and PR curves measured for each client after each federated round are summarized in Table IV and Table V accordingly, due to page limit constraints. As indicated, even after the first federated round, each client produces very similar ROC curves to the centralized model, delivering almost 100% in AUC. For the AUC-PR, the increase in performance after each subsequent round is more visible, which supports the argument that utilizing only the ROC metric is not enough. Especially for Client 2, Precision drops more abruptly for higher Recall values after the first round but improves after all federated rounds are completed.



(a) ROC curve. (b) ROC, close-up.



(c) PR curve.

Fig. 3: ROC curve and PR curve of the centralized model.

Figure 3 depicts the ROC and PR curves measured in the centralized setting. Due to page limit constraints, we summarize the federated ROC and PR results in Table IV and Table V. For the AUC-PR, the increase in performance after each subsequent round is more visible, which supports the argument that utilizing only the ROC metric is insufficient. Especially for Client 2, Precision drops more abruptly for higher Recall values after the first round but improves at the end.

In the centralized setting, we measure the time spent on training and testing. We compare it to the duration of the whole FL process, which consisted of 3 federated rounds. Each client needs approximately 420 seconds per round for training and 3.5 seconds for testing. The training time of a client is shorter than the training time of the centralized training due to the five times larger dataset used in the latter setting. However, five clients utilized the entire dataset for three federated rounds, constituting 150 training epochs versus 50 training epochs on the full dataset in centralized learning. Still, it takes less time, thanks to the distribution of the learning process. Considering the demonstrated high performance and time efficiency, combined with the security and privacy-preserving nature of FL, the proposed federated approach to the semi-supervised IDS in IIoT systems proved to be a promising alternative to centralized learning. However, even though both centralized and federated scenarios deliver high AUC values, the detection performance would depend on the chosen anomaly threshold in a real-world application. There are various approaches to choosing a threshold, for example, sorting all anomaly scores and placing the top k of them into the anomalous class. Another solution is to evaluate the ROC curve and set a threshold that delivers the most suitable TPR and FPR. For example, by zooming into the ROC plot of the centralized model as shown in Figure 3b, it is visible that higher TPR results in higher FPR.

V. Conclusions

In this paper, we addressed the concerns for IIoT network security. This context can significantly benefit from anomaly-based IDSs to detect unusual activity in the traffic and, potentially, zero-day attacks. Using semi-supervised approaches, industrial organizations can utilize massive unlabeled traffic data produced by IIoT devices and enhance performance by labeling a fraction of the data. In this paper, we summarized the theoretical justification of the approach, proposed concrete frameworks for federated semi-supervised anomaly detection using the promising DeepSAD model, introduced methods for handling the categorical data, and measured the framework performance in various configurations. Furthermore, we compared the framework's performance in federated and centralized scenarios. The results are promising, with very high precision in detecting DoS attacks in federated settings. Nevertheless, we plan future improvements to cover the following limitations: 1) In this work, we consider one semi-supervised setting (Label-distributed Federated SSL with Labels-at-Client scenario), but is essential to study the performance of FL under **different label distributions**; 2) The case study experiments were simulated in

a single-machine. We plan to review the proposed framework in a **real physical network**, for example, in a small-scale IIoT testbed; 3) Anomaly detection only considered DoS attacks. For real-work applications, it is essential to extend the framework to the **multi-anomalies classification**, including sophisticated attacks. 4) It is crucial to explore the **security risks of FL** and explore the extension of the proposed framework with additional security and privacy-preserving methods such as encryption.

REFERENCES

[1] H. Boyes *et al.*, "The industrial internet of things (IIoT): An analysis framework," *Computers in Industry*, vol. 101, pp. 1–12, 2018.

[2] L. D. Xu *et al.*, "Internet of Things in Industries: A Survey," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.

[3] X. Jiang *et al.*, "An experimental analysis of security vulnerabilities in industrial iot devices," *ACM Transactions on Internet Technology (TOIT)*, vol. 20, no. 2, pp. 1–24, 2020.

[4] Kaspersky ICS CERT, "Threat landscape for industrial automation systems. Statistics for H2 2021," https://ics-cert.kaspersky.com/publications/reports/2022/03/03/threat-landscape-for-industrial-automation-systems-statistics-for-h2-2021/, 2022, accessed: 19-06-2022.

[5] M. Conti *et al.*, "A Survey on Industrial Control System Testbeds and Datasets for Security Research," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2248–2294, 2021.

[6] X. Yang *et al.*, "Security vulnerabilities in lorawan," in *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE, 2018, pp. 129–140.

[7] A. Dorri *et al.*, "Towards an optimized blockchain for iot," in *2017 IEEE/ACM Second International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE, 2017, pp. 173–178.

[8] A. Rullo *et al.*, "A game of things: Strategic allocation of security resources for iot," in *2017 IEEE/ACM Second International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE, 2017, pp. 185–190.

[9] S. H. Hashemi *et al.*, "World of empowered iot users," in *2016 IEEE first international conference on internet-of-things design and implementation (IoTDI)*. IEEE, 2016, pp. 13–24.

[10] A. M. da Silva Cardoso *et al.*, "Real-time ddos detection based on complex event processing for iot," in *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE, 2018, pp. 273–274.

[11] G. D. Putra *et al.*, "Towards scalable and trustworthy decentralized collaborative intrusion detection system for iot," in *2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE, 2020, pp. 256–257.

[12] F. De Vita *et al.*, "A novel data collection framework for telemetry and anomaly detection in industrial iot systems," in *2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE, 2020, pp. 245–251.

[13] H. G. Abreha *et al.*, "Federated Learning in Edge Computing: A Systematic Survey," *Sensors*, vol. 22, no. 2, 2022.

[14] B. McMahan *et al.*, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, A. Singh and J. Zhu, Eds., vol. 54. PMLR, 20–27 Apr 2017, pp. 1273–1282.

[15] D. J. Beutel *et al.*, "Flower: A Friendly Federated Learning Research Framework," *arXiv preprint arXiv:2007.14390*, 2020.

[16] R. T. I. Inc, "Industrial Internet of Things," https://info.rti.com/hubfs/docs/Industrial_IoT_FAQ.pdf, 2015, accessed: 20-07-2022.

[17] E. Byres *et al.*, "The myths and facts behind cyber security risks for industrial control systems," 2004.

[18] J. Sengupta *et al.*, "A Comprehensive Survey on Attacks, Security Issues and Blockchain Solutions for IoT and IIoT," *Journal of Network and Computer Applications*, vol. 149, 2020.

[19] M. G. S. Murshed *et al.*, "Machine Learning at the Network Edge: A Survey," *ACM Computing Surveys*, vol. 54, no. 8, pp. 1–37, 2022.

[20] A. L. Buczak and E. Guven, "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2015.

[21] N. Goernitz *et al.*, "Toward Supervised Anomaly Detection," *Journal of Artificial Intelligence Research*, vol. 46, no. 1, pp. 235–262, 2013.

[22] A. Aldweesh *et al.*, "Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues," *Knowledge-Based Systems*, vol. 189, 2020.

[23] J. E. v. Engelen and H. H. Hoos, "A survey on semi-supervised learning," *Machine Learning*, vol. 109, no. 2, pp. 373–440, 2020.

[24] N. Rieke *et al.*, "The future of digital health with federated learning," *NPJ Digital Medicine*, vol. 3, no. 1, p. 119, 2020.

[25] D. Verma *et al.*, "Federated Learning for Coalition Operations," *arXiv preprint arXiv:1910.06799*, 2019.

[26] Q. Yang *et al.*, "Federated learning: Privacy and incentive," *Federated Learning*, 2020.

[27] D. Xu *et al.*, "Edge Intelligence: Architectures, Challenges, and Applications," *arXiv preprint arXiv:2003.12172*, 2020.

[28] M. Aledhari *et al.*, "Federated Learning: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Access*, vol. 8, pp. 140 699–140 725, 2020.

[29] Y. Zhao *et al.*, "Multimodal federated learning on iot data," in *2022 IEEE/ACM Seventh International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE, 2022, pp. 43–54.

[30] T. Yu *et al.*, "Learning context-aware policies from multiple smart homes via federated multi-task learning," in *2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE, 2020, pp. 104–115.

[31] Y. Gao *et al.*, "Linklab: A scalable and heterogeneous testbed for remotely developing and experimenting iot applications," in *2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE, 2020, pp. 176–188.

[32] Y. Jin *et al.*, "A Survey towards Federated Semi-supervised Learning," *arXiv preprint arXiv:2002.11545*, 2020.

[33] W. Jeong *et al.*, "Federated Semi-Supervised Learning with Inter-Client Consistency & Disjoint Learning," *arXiv preprint arXiv:2006.12097*, 2020.

[34] M. Zolanvari *et al.*, "Machine Learning-Based Network Vulnerability Analysis of Industrial Internet of Things," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6822–6834, 2019.

[35] M. A. Teixeira *et al.*, "SCADA System Testbed for Cybersecurity Research Using Machine Learning Approach," *Future Internet*, vol. 10, no. 8, 2018.

[36] M. Zolanvari *et al.*, "WUSTL-IIOT-2021 Dataset for IIoT Cybersecurity Research," https://www.cse.wustl.edu/~jain/iiot2/index.html, Washington University in St. Louis, USA, October 2021, accessed: 2022-05-21.

[37] C. Guo and F. Berkhahn, "Entity Embeddings of Categorical Variables," *arXiv preprint arXiv:1604.06737*, 2016.

[38] Y. Seth, "A neural network in pytorch for tabular data with categorical embeddings," 2018, accessed: 2022-05-21. [Online]. Available: https://yashuseth.wordpress.com/2018/07/22/pytorch-neural-network-for-tabular-data-with-categorical-embeddings/

[39] L. Ruff *et al.*, "Deep Semi-Supervised Anomaly Detection," in *International Conference on Learning Representations*. OpenReview.net, 26 Apr–1 May 2020. [Online]. Available: https://openreview.net/forum?id=HkgH0TEYwH

[40] J. Davis and M. Goadrich, "The Relationship Between Precision-Recall and ROC Curves," in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML '06. New York, NY, USA: Association for Computing Machinery, 25 Jun – 29 Jun 2006, p. 233–240.

[41] R. Liaw *et al.*, "Tune: A Research Platform for DistributedModel Selection and Training," *arXiv preprint arXiv:1807.05118*, 2018.

[42] T. Nguyen *et al.*, "DÏot: A federated self-learning anomaly detection system for iot," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, 7-9 Jul 2019, pp. 756–767.