Federated Learning for Network Anomaly Detection in a Distributed Industrial Environment

A Dehlaghi-Ghadim[1,2,*], T Markovic[2,*], M Leon[2,*], D Söderman[3], and P E Strandberg[3]

[1]Research Institute of Sweden (RISE), Västerås, Sweden
[2]Mälardalen University, Västerås, Sweden
[3]Westermo Network Technologies AB, Västerås, Sweden
[*]The first three authors contributed equally to this research.

Abstract

Industrial control systems have been targeted by numerous cyber attacks over the past few decades which causes different problems related to data privacy, financial losses and operational failures. One potential approach to detect these attacks is by analyzing network data using machine learning and employing network anomaly detection techniques. However, the nature of these systems often involves their geographical dispersion across multiple zones, which poses a challenge in applying local machine learning methods for detecting anomalies. Additionally, there are instances where sharing complete operational data between different zones is restricted due to security concerns. As a result, a promising solution emerges by implementing a federated model for anomaly detection in these systems. In this study, we investigate the application of machine learning techniques for anomaly detection in network data, considering centralized, local, and federated approaches. We implemented the local and centralized methods using several simple machine-learning techniques and observed that Random Forest and Artificial Neural Networks exhibited superior performance compared to other methods. As a result, we extended our analysis to develop a federated version of Random Forest and Artificial Neural Network. Our findings reveal that the federated model surpasses the performance of the local models, and achieves comparable or even superior results compared to the centralized model, while it ensures data privacy and maintains the confidentiality of sensitive information.

Keywords: Network Anomaly Detection, Machine Learning, Federated Learning, Random Forest, Artificial Neural Network.

I Introduction

Modern industrial communication systems may be composed of a core network of trusted switches and routers developed in accordance with best industrial practices [35]. For each router in the core network, there may be tens of end devices. In traditional, off-line, industrial networks, the set of devices and the topology of the factory, is often static, meaning that the set of expected traffic flows is well defined in advance. However, with the rise of Industry 4.0 and Internet of Things (IoT) the end devices may become even more numerous, less expensive, and likely more vulnerable to cyber attacks [35]. Such a vulnerability may escalate into events such as the Colonial Pipeline ransomware attack, that halted pipeline operations, or even to death of humans [21]. Less dramatic, but almost as common problems occur from human error, e.g. misconfiguration [25]. All these problems can be seen as anomalies in the network traffic, i.e. some deviations from the normal behaviour [4]. One possible mitigation is employment of anomaly detection techniques. By monitoring network traffic, alarms may be raised an attack is ongoing or when human misconfiguration is detected.

Utilizing Machine Learning (ML) [33] and Deep Learning (DL) [18] in intrusion detection systems showed good performance for detecting anomalies in network data. The ongoing Artificial Intelligence (AI) chip race is expected to drive down the cost of AI/ML computation, making it more accessible for various applications [24]. This opens up possibilities for integrating AI/ML-enhanced anomaly detection into industrial communication systems, including switches, routers, and other devices. Consequently, researchers have focused on developing ML-based solutions [13, 29] to detect anomalies in the network traffic. However, these solutions have certain drawbacks, such as the need for centralized data storage for ML models training and the security risks associated with transmitting data from local nodes to a central server. Additionally, network bandwidth limitations and latency issues may arise during data transmission to the server [22]. To address these limitations, Federated Learning (FL) has emerged as a promising approach, enabling machine learning on decentralized data sources while ensuring privacy, security, and efficiency [17, 12]. Furthermore, FL has demonstrated remarkable results in the domain of intrusion detection. For instance, Tang et al. [32] achieved accuracy levels nearly equivalent to those of centralized deep learning models when applied to the CIC-IDS-2017 network intrusion detection dataset. Moreover, Chen et al. [5] introduced the Federated Learning-based Attention Gated Recur-

rent Unit (FedAGRU), which enhances detection accuracy by approximately 8%, highlighting the potential of FL in this domain.

In this paper, we explore FL for network anomaly detection in a distributed industrial network. To achieve this, we used the Westermo network traffic data set [31], that was generated by intentionally introducing various network anomalies into a simulated factory testbed, including network cyber attacks and network switch misconfigurations. The network traffic was captured at multiple locations within the network. The goal of this work is to detect these anomalies using different ML algorithms placed on different locations in the network. The work focuses on the supervised ML algorithms [20], including: Logistic Regression, Support Vector Machine, Artificial Neural Network, K-Nearest Neighbours, Decsion Tree and Random Forest.

The contributions of this paper cover implementing and comparing three different approaches using different ML algorithms for anomaly detection and classification in distributed network data:

- a centralized approach - where all the data was transmitted to a central node for analysis.
- a local approach - where each switch has the responsibility of detecting the anomalies in its own data.
- a FL approach - where the local switches collaborate to create a global model.

To the best of authors knowledge the used dataset is an unique example of open industrial data where the network data was recorded on multiple locations in a distributed system, which makes it perfect to evaluate FL approaches.

Our findings indicate that the FL approach achieved comparable or even superior performance to the centralized model while also ensuring data privacy and security. This implies that the FL approach holds promise for detecting network anomalies effectively without compromising sensitive information.

The remaining sections of this paper are structured as follows: Section II presents the methodology employed in this research, encompassing details about the dataset, the experiments conducted, and the experimental settings. Section III presents the results obtained from our experiments. In Section IV, we discuss the results, and outline future directions. Finally, Section V concludes the paper.

II Methodology

This section presents the dataset used, machine learning and federated learning algorithms, as well as experimental settings.

A Dataset

The Westermo network traffic data set[1] [31], was created to simulate an industrial communication system and gather data from multiple nodes in the network simultaneously. The network consisted of six network switches that acted as the core of the network. These switches utilized the widely used redundancy protocol, called Rapid Spanning Tree Protocol (RSTP). To further enhance the simulation, the Industrial Control System SIMulator (ICSSIM), [8], was employed. ICSSIM was designed to emulate a bottle-filling factory and consisted of two Programmable Logic Controller (PLC) nodes, one Human Machine Interface (HMI), and a simulator for the physical world.

Data collection was made repeatable by utilizing a script within the Westermo test framework [30]. Initially, a known normal state was configured. Subsequently, data recording commenced in three nodes by redirecting network traffic from these devices to a laptop running the test framework. The script then iterated through various misconfigurations and attacks against the devices for approximately 90 minutes. The events included:

- *Bad-Misconf:* misconfigured IP addresses, such as setting an IP to 198.134.18.7 instead of 198.18.134.7 to simulate human error.
- *Bad-Misconf-Duplication:* IP duplication to simulate human error.
- *Bad-MITM:* a man-in-the-middle (MITM) attack that intercepted and modified packet contents between a PLC and the HMI.
- *Bad-Portscan1:* a single device port scan using NMAP.
- *Bad-Portscan2:* Multiple device port scan with NMAP.
- *Bad-SSH:* SSH password guessing.
- *Good-SSH:* Login over SSH.

Every event was implemented within the test framework, except the MITM attack that was conducted using ICSSIM from a separate Raspberry Pi. Data was collected from three switch nodes: Right, Bottom, and Left. Upon completing the data collection process, three PCAP files were generated.

The final dataset consists of three CSV-formatted files, one for each switch node, containing network flow data, generated by analyzing the PCAP files using the ICSFlowGenerator tool [7]. This tool examines the raw data and computes various characteristics of network flows, including flow network addresses, number of sent and received packets, packet size, duration, network protocol, average payload size, and various TCP features such as packet delays, TCP flags statistics, TCP header information, etc. This process resulted in 60 features for each instance in the dataset, where each instance represents one network flow. Additionally, the CSV files contain four different labels following two different labeling strategies: NST [10] and IT [16], for both binary and multiclass classification. The resulting dataset is obtained by

[1] https://github.com/westermo/network-traffic-dataset

processing over 1.8 million network packets to 48657 network flows and is presented in Table 1, which outlines the class distribution of the generated datasets.

B Dataset pre-processing

During the dataset preprocessing phase, we performed the following steps:

1. *Removing features:* The IP and MAC address columns in the datasets were removed because using this data makes anomaly detection a trivial task, as most anomalies are initiated from one attacker node. Additionally, 6 other time-related columns were removed. This process resulted in 48 features.

2. *Feature transformation:* Categorical columns were converted into an array of binary columns using the one-hot encoding technique. This process resulted in 53 features in total.

3. *Data cleaning:* Missing values in the dataset (e.g., TCP features for non-TCP network flows) were filled with zero.

4. *Data normalization:* Input variables were normalized. All input variables are either binary values or real numbers. Binary values remained unchanged, while real numbers were normalized in the range [0, 1] using the Min-Max normalization technique with respect to the training set.

5. *Label selection:* We selected NST labeling strategy by using 'NST-B-Label' for anomaly detection and 'NST-M-Label' for anomaly classification

6. *Label transformation:* We changed the class of 'Good-SSH' to the 'Normal' class, as those network flows are not indeed anomalies.

C Machine Learning Algorithms

In this paper, we are using six supervised ML algorithms to perform the task of network attack detection and classification. This subsection briefly describes those ML algorithms:

1) Logistic regression (LR): LR [23] is a statistical technique used to analyze the relationship between a characteristic of interest and a set of independent variables. It predicts the probability of an input belonging to a specific class by fitting a logistic function to the data. The logistic function maps the features to a probability score of 0 to 1. Based on this probability, LR assigns the input to the class with the highest likelihood, enabling it to perform binary and multi-class classification.

2) Support Vector Machine (SVM): SVM [1] is an algorithm that finds a hyperplane in a high-dimensional feature space, which maximizes the separating margin between different classes. The higher-dimensional feature space is created by employing kernel functions to transform the data.

3) Artificial Neural Network (ANN): ANN [29] is composed of interconnected nodes, known as neurons, which are arranged into layers. In this network, each neuron receives inputs which are multiplied by the weights assigend to that neuron, and then applies an activation function to generate an output. This output is used to detect the target classes. The complexity of the problem determines the number of hidden layers and neurons of each layer that can be present in an ANN.

4) K-Nearest Neighbors (KNN): KNN [6] is a distance-based method that predicts the target class by considering the K closest training examples in the feature space. The algorithm uses a majority vote of the labels or of these nearest neighbors to perform classification. The choice of K determines the balance between bias and variance in the model. Smaller values lead to more complex models that may overfit the data, while larger values introduce more bias but reduce the risk of overfitting. The distance metric used to calculate proximity is another parameter to tune in this algorithm.

5) Decision Tree (DT): DT [14] is an ML algorithm formed by decision nodes and leaf nodes. The values of the features from the dataset are compared with a threshold at the decision nodes, with each possible outcome resulting in a branch that can lead either to another decision node or to a leaf node that contains a final prediction. Different splitting rules [34] can be used to construct a DT and in this paper we considered two mostly used ones: gini and entropy.

6) Random Forest (RF): RF [28] is composed of multiple DTs, where each tree uses a random subset from the training set. The number DTs is a hyper-parameter of RF. In this paper, the final prediction of RF is calculated by aggregating the predictions of the different DTs using the simple voting technique, which takes a majority vote as the predicted class.

D Federated Learning

The goal of the paper is to evaluate how a FL setup influences the performance of ML algorithms. In a FL setup, individual models that are trained on individual clients are merged in a centralized server in order to share the gained knowledge. Three ML algorithms that have the best performance on individual clients (refer to Section B) are RF, KNN, and ANN. Since KNN is not a suitable choice for FL (it keeps all data in the memory which breaks FL privacy paradigm), we decided to develop RF and ANN in a federated setup.

1) RF FL: In this paper, we are using an FL framework for network attack detection and classification based on RF, which is proposed in [19]. The main idea of the framework is to train independent RFs on clients using the local data, merge independent models into a global one on the server and send it back to the clients for further use. The independent models can be merged using four different methods. The first two methods sort DTs per RF and select the best ones from each RF based on the accuracy

Table 1: Statistics of network flows in the Right, Bottom, and Left dataset.

	Right		Bottom		Left		All	
	no	**%**	**no**	**%**	**no**	**%**	**no**	**%**
Normal	3464	73.4%	29159	82.7%	4025	47.2%	36,648	75.3%
Bad-Misconf	261	5.5%	2103	5.9%	313	3.7%	2677	5.5%
Bad-Misconf-Duplication	311	6.6%	2453	6.9%	367	4.3%	3131	6.4%
Bad-MITM	102	2.2%	504	1.4%	102	1.2%	708	1.5%
Bad-Portscan1	9	0.2%	29	0.1%	229	2.7%	267	0.5%
Bad-Portscan2	299	6.3%	397	1.1%	1483	17.4%	2179	4.5%
Bad-SSH	265	5.6%	751	2.1%	1952	22.9%	2968	6.1%
Good-SSH	9	0.2%	8	0.03%	62	0.7%	79	0.2%
Sum	4720	100.0%	35404	100.0%	8533	100.0%	48657	100.0%

(Sorting DTs per RF based on Accuracy - S_DTs_A) or weighted accuracy (Sorting DTs per RF based on Weighed Accuracy - S_DTs_WA). The remaining two methods assemble all DTs from all independent RFs and select the best ones based on the accuracy (Sorting All DTs based on Accuracy - S_DTs_A_All) or weighted accuracy (Sorting All DTs based on Weighed Accuracy - S_DTs_WA_All).

2) ANN FL: Our approach involves training local classifiers on individual devices using the data stored locally. Once the local training is complete, the local models are transmitted to the central server. At the central server, we perform model aggregation by combining the models from different devices. We create an ensemble model that aggregates the outputs of the local models. To achieve this, we employ three different aggregation methods. The first aggregation method, ANN_FL, calculates the average prediction of the local models' outputs. This method takes the mean of the predictions generated by each local model. The second method, called ANN_FL_Weight, utilizes weighted averaging, considering the relative sizes of the trained datasets on each local device. The weights, $w_1, w_2, ..., w_n$, assigned to the models are based on the proportions of the trained data on each device. The third method, called ANN_FL_Rank, also employs weighted averaging, but the weights are assigned based on the rank of the local models. Each local model is assigned an integer rank weight (w_i) ranging from 1 to n, with the model created by the smallest dataset assigned weight 1 and the model created by the largest dataset assigned weight n.

E Experimental Settings

To explore the effectiveness of different approaches for Anomaly Detection (AD) and Anomaly Classification (AC), in other words binary (2 classes) and multi-class classification (7 classes), on network data, three distinct experiments are conducted: centralized approach, local approach, and FL approach.

1) Experiment 1 - centralized approach: The fundamental assumption is that the central node performs AD/AC by considering all the information received from the nodes. This process requires merging three distinct datasets into a single entity, which is then divided into training, validation, and testing sets in proportions of 70-10-20%, respectively. The training dataset is used to develop an AD/AC model, while the validation dataset is used to fine-tune hyperparameters of different ML algorithms. Finally, the performance of these models is evaluated using the test dataset.

2) Experiment 2 - local approach: In this approach, the individual nodes are responsible for conducting AD/AC using models generated using their local data. To simulate this scenario, we partition the three datasets into training, validation, and testing subsets, preserving the same proportions as in Experiment 1. We independently trained ML models using each training subset and used the validation subset to determine the optimal hyperparameters. Subsequently, we assessed the performance of these models on their respective test subsets, as well as on the test subsets of other nodes.

3) Experiment 3 - FL approach: In this approach, the server takes on the responsibility of creating an AD/AC model without directly receiving training data from the local nodes. The approach combines the individual models trained by the nodes in Experiment 2 to develop a centralized model. To evaluate the effectiveness of the FL model, we assess its performance using the three distinct test sets from each local node (Right, Bottom and Left test set), as well as using the all test dataset together (All test set).

To improve the estimation of model performance, optimize data utilization, and mitigate bias, we employed 5-fold cross-validation. This approach involves creating separate models for each fold and obtaining average performance results across the folds [26]. Additionally, our datasets consist of imbalanced data with varying instances of anomalies and normal classes. To tackle this challenge, we measure accuracy in percentage, and F1-score, recall, and precision in $[0, 1]$ range for all experiments. Among these metrics, we primarily utilize the F1-score for comparison, as it is based on precision and recall and exhibits a similar correlation to accuracy.

III Results

In this section, we present the results from the three experiments: centralized approach, local approach, and FL approach, for both AD and AC.

A Experiment 1 - centralized approach

The results of Experiment 1, where the ML algorithms are tested on the entire dataset, are shown in Table 2. RF obtained the best performance, with F1-score around 0.85 for AD and 0.81 AC. Then, very closely, ANN and KNN obtained a F1-score of between 0.82 and 0.84 for AD, and 0.78 to 0.80 for AC. Finally, we can see how the worst performance is obtained by SVM and LR for both problems. An interesting remark can be made for LR, which increases its F1-score by 0.05 from AD to AC. All other algorithms have a lower F1-score when solving AC in comparison to AD. RF and ANN are selected to be used in a FL setup as the best performing and most suitable algorithms.

B Experiment 2 - Local approach

The results of Experiment 2, where ML algorithms are trained and tested locally on the clients, using their own subsets, are shown in Fig. 1. The same pattern between AD and AC can be noticed since all the algorithms have a similar performance independently of the problem to be solved. For instance, KNN training with the Right subset and testing on the Bottom subset has poor performance for both AD and AC. The only algorithm that does not have a correlation is LR, which performs much better for AD than for AC.

Additionally, we can see similarities between the algorithms. The first group, formed by Decision tree-based algorithms (DT and RF), performs in the same way. We can see how, independently on which data we train or test, we can have satisfactory performance. This means that the algorithm can retrieve information from subsets with a small amount of data (Left and Right) and still perform AD or AC on the big subset (Bottom). In contrast, the second group, formed by KNN, SVM, and ANN, cannot obtain any knowledge from the Left and Right subsets that can be used on the Bottom subset. This last statement is also true when performing AC with LR.

C Experiment 3 - FL approach

The results of ML algorithms that are implemented in a FL setup are presented on Fig. 2 and further discussed in this subsection. Each FL algorithm from the server is compared to its own client versions, on three separate testing sets and the entire testing set.

1) FL RF: Results for RF are presented on Fig. 2a and 2b, for AD and AC respectively. For both AD and AC, we can notice that RF from the server outperforms all three client versions when it is tested on the entire testing set. When it is tested on the subsets from clients, we can see that it can reach a performance that is very close to the performance of the client model on the subset of that specific client. With respect to hyper-parameters that were used on the server, the methods that assemble all DTs from all independent RFs, and then select the best ones (S_DTs_A_All and S_DTs_WA_All) had the best performance for most of the independent runs (5 folds). When it comes to splitting methods and the number of DTs, entropy was the most successful splitting rule, while the final number of DTs used in the server varied from 10 to 90 for AD and from 20 to 35 for AC.

2) FL ANN: Results for ANN are presented on Fig. 2c and 2d, for AD and AC respectively. For AD, it is evident that the FL approach outperforms the single clients across most of datasets, particularly when considering the aggregated test set (All). However, there is a slight decrease in performance observed on the Bottom test set compared to the model from Bottom client. Similarly, in the AC task, the FL approach consistently achieves better or comparable performance to the local models. Notably, a significant improvement is observed when using the FL approach on the aggregated test set. As mentioned before, the potential aggregation approaches included in this work are ANN_FL, ANN_FL_Weight, and ANN_FL_Rank. From them, ANN_FL_Weight outperformed other methods on most of the independent runs (5 folds).

IV Discussion

In this section, we discuss the results by comparing the centralized, local and FL approaches. Additionally, potential future directions are presented.

A Comparison of FL approach with the centralized and local approach

The results of RF and ANN for the different approaches are given in Table 3. For RF, we can see that the results of centralized approach are better than for the local one. This means that by training on a single subset, we gain less knowledge than by training on the entire dataset. However, by using FL approach, where the training is done on each subset separately, same as in the local approach, but the knowledge is combined, the performance of the algorithm improves by 0.035 for AD and by 0.026 for AC, compared to the centralized approach. This improvement is expected since, even thought independent RFs are trained on the local data, only the best DTs from them are kept in the global model.

In the case of ANN, it is evident that the performance of centralized approach in AD surpasses the performance of local models. This comparison indicates that when conducting AD with partial observations of network traffic, there is a potential for performance degradation. However, the performance of FL approach demonstrates that FL can mitigate these performance losses, although it is less effective than centralized AD when considering ANN. Nonetheless, FL achieves comparable accuracy in AC when employing aggregated classification and leveraging shared models.

Table 2: Results of centralized ML models AD and AC using the entire dataset. Three best algorithms are shown in boldface.

	AD				AC			
ML	**Accuracy (%)**	**F1-score**	**Precision**	**Recall**	**Accuracy (%)**	**F1-score**	**Precision**	**Recall**
LR	75.4814	0.6494	0.5697	0.7548	78.6752	0.7030	0.6393	0.7868
SVM	75.4650	0.6501	0.6560	0.7546	75.4773	0.6494	0.5738	0.7548
ANN	**85.1409**	**0.8273**	**0.8703**	**0.8514**	**84.7031**	**0.784**	**0.7454**	**0.847**
KNN	**85.9506**	**0.8450**	**0.8612**	**0.8595**	**85.0731**	**0.8055**	**0.7760**	**0.8507**
DT	81.0675	0.8063	0.8038	0.8107	79.5405	0.7862	0.7779	0.7954
RF	**86.9125**	**0.8545**	**0.8762**	**0.8691**	**86.2836**	**0.8137**	**0.7876**	**0.8628**

Algorithm	**Train**	**AD**				**AC**			
		Test				**Test**			
		Right	**Bottom**	**Left**	**All**	**Right**	**Bottom**	**Left**	**All**
LR	**Right**	0.648	0.750	0.372	0.668	0.703	0.227	0.582	0.370
	Bottom	0.658	0.756	0.385	0.676	0.624	0.744	0.310	0.649
	Left	0.717	0.650	0.531	0.639	0.704	0.195	0.562	0.348
SVM	**Right**	0.857	0.237	0.836	0.427	0.710	0.184	0.485	0.330
	Bottom	0.624	0.744	0.312	0.650	0.624	0.744	0.310	0.649
	Left	0.858	0.237	0.835	0.427	0.704	0.185	0.532	0.334
ANN	**Right**	0.773	0.284	0.810	0.442	0.779	0.217	0.791	0.395
	Bottom	0.752	0.730	0.736	0.727	0.763	0.783	0.665	0.760
	Left	0.741	0.240	0.800	0.406	0.779	0.219	0.791	0.398
KNN	**Right**	0.830	0.214	0.708	0.379	0.812	0.167	0.694	0.323
	Bottom	0.777	0.824	0.741	0.811	0.759	0.803	0.679	0.778
	Left	0.756	0.157	0.888	0.379	0.690	0.123	0.827	0.320
DT	**Right**	0.863	0.792	0.546	0.781	0.825	0.650	0.496	0.654
	Bottom	0.807	0.781	0.873	0.798	0.778	0.770	0.759	0.771
	Left	0.608	0.812	0.893	0.819	0.362	0.782	0.830	0.761
RF	**Right**	0.863	0.631	0.619	0.665	0.824	0.520	0.600	0.560
	Bottom	0.807	0.832	0.884	0.847	0.787	0.809	0.831	0.812
	Left	0.404	0.801	0.895	0.793	0.506	0.780	0.835	0.769

Figure 1: F1-score of local ML algorithms when training and testing on different clients for AD and AC problems. Better values are shown with a darker blue.

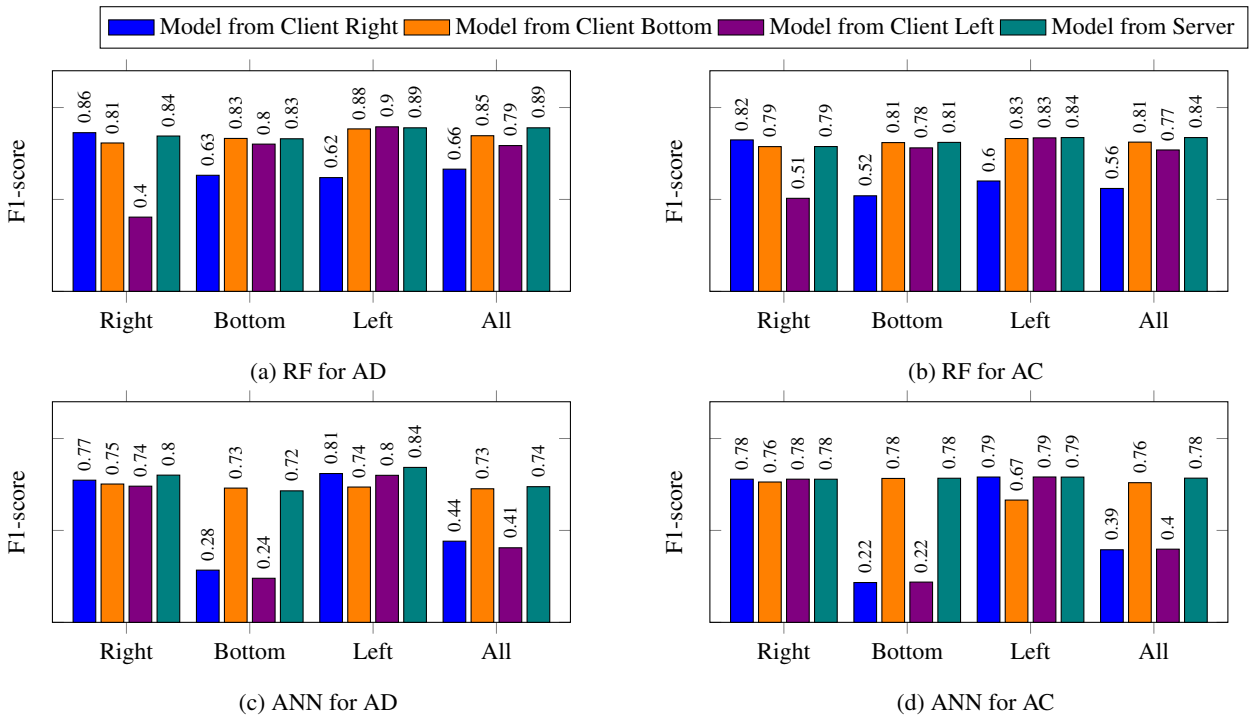
Notably, the FL version of RF consistently outperforms ANN, as evidenced by their performance on the entire dataset: 0.89 for AD compared to 0.74, and 0.84 for AC compared to 0.78.

Additionally, the performance of FL algorithms on different classes of anomalies, as illustrated in Figure 3, reveals interesting findings. Both RF and ANN have a very good performance for detecting three out of six classes (0:Normal, 5:Bad-Portscan2, and 6:Bad-SSH). On the other hand, both of them demonstrate poor performance in detecting misconfiguration anomalies, however, RF shows some tiny success compared to ANN that has F1-score equal to 0. A similar behavior can be noticed with the class 4:Bad-Portscan1, due to the small number of instances of this class in the dataset. Furthermore, ANN exhibits weak performance in detecting MITM attacks, whereas RF is able to detect it with a very good performance.

It is important to mention that the remarkable performances of FL approach are achieved while ensuring data security and privacy since this approach does not involve transferring the entire training set to the centralized location.

B Future directions

This paper uses a network traffic dataset, which contains Operational Technology (OT) network emulation. However, considering the future of industrial networks, it is expected that there will be a combination of traditional OT devices, IT-type network functionalities, heterogeneous services, and communication protocols over the edge-fog-cloud spectra, as well as a larger number of potentially inexpensive nodes [35, 2, 9, 15]. Additionally, network segmentation is a commonly used counter-measure to address cybersecurity problems [11, 27]. Therefore, a first topic for future work could cover the creation of improved datasets that will include larger, heterogeneous, and segmented networks to represent real-world scenarios better. Additionally, the datasets can be improved by introducing more attacks, and in particular,



(a) RF for AD

(b) RF for AC

(c) ANN for AD

(d) ANN for AC

Figure 2: F1-score of local ML models on clients and global ML on the server on different test sets

Table 3: Comparison of F1-score of different approaches on the entire testing set. The best approach per algorithm and problem is shown in boldface.

Problem:	**AD**					**AC**				
Approach:	**Centralized**	**Local**			**FL**	**Centralized**	**Local**			**FL**
Algorithm		**Min**	**Avg**	**Max**			**Min**	**Avg**	**Max**	
RF	0.855	0.665	0.768	0.847	**0.890**	0.814	0.560	0.714	0.812	**0.840**
ANN	**0.827**	0.406	0.525	0.727	0.738	**0.784**	0.395	0.518	0.760	**0.784**

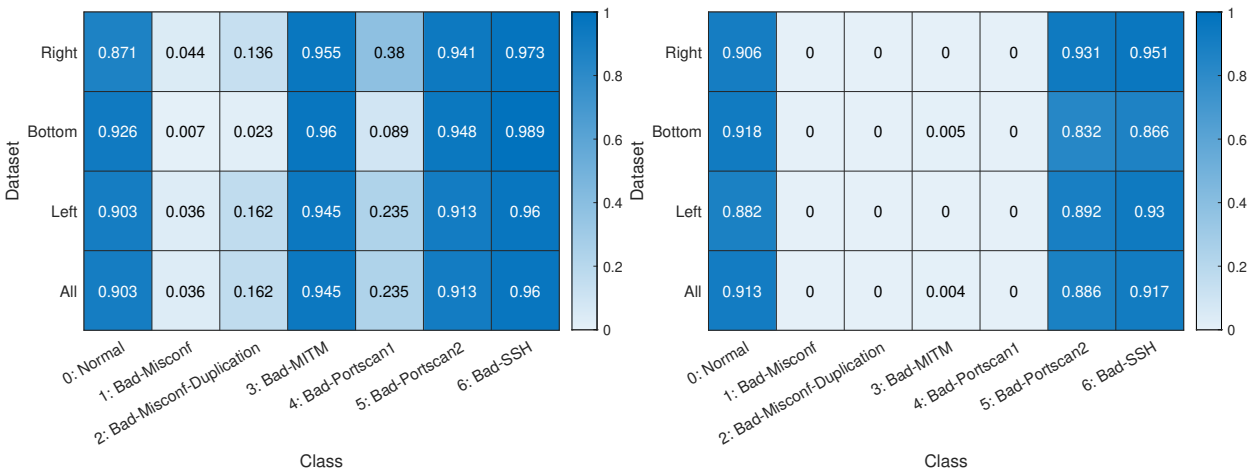
more network misconfigurations [25], and other human errors that can result in network anomalies. To the best of our knowledge, human errors are not well covered in network traffic data sets other than Westermo's.

Secondly, our results show that ML was not very effective in detecting misconfiguration. A possible future direction would be to use hybrid intrusion detection combining rule-based and learning intrusion detection systems [3].

Thirdly, a promising direction to extend this work is to explore the feasibility of deploying federated AI/ML-powered anomaly detection directly within routers. Both the trends of less expensive AI chips and the advances in network with a changing edge-fog-cloud spectra would impact this line of research [35, 2, 9, 24]. By investigating the implementation of federated AI/ML models at the network edge, fog, and/or cloud, researchers could explore trade-offs between computational power, storage possibilities, as well as network latency and load.

V Conclusions

This paper investigates utilizing ML techniques for AD and AC in industrial communication systems, providing a comparative analysis of centralized, localized, and FL approaches. To achieve this, we conducted experiments Westermo network traffic data set containing various network anomalies, such as cyber-attacks and switch misconfigurations. Our study involved implementing centralized and local ML models with six different ML algorithms, showing that RF and ANN outperformed the other algorithms, demonstrating superior performance for AD and AC. Additionally, we observed that employing only the localized data from individual nodes in the network significantly decreased the effectiveness of models when compared to the centralized approach. To prioritize data privacy and confidentiality of sensitive information while maintaining performance, we investigated the FL approach by harnessing decentralized data sources and collaborative model development. Specifically, we developed a federated version of RF and ANN. Our research revealed that the federated model achieved compa-



(a) RF (b) ANN

Figure 3: F1-score of FL RF and FL ANN on the different classes for the entire testing set.

rable or even superior performance when compared to the centralized model, all while ensuring the preservation of data privacy and security. This implies that the FL approach holds promise for detecting network anomalies effectively without compromising sensitive information.

Acknowledgment

This work has been partially supported by Westermo Network Technologies AB, and the H2020 ECSEL EU projects Intelligent Secure Trustable Things (InSecTT) and Distributed Artificial Intelligent System (DAIS). InSecTT (www.insectt.eu) has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 876038 and DAIS (https://dais-project.eu/) has received funding from the ECSEL JU under grant agreement No 101007273. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Austria, Sweden, Spain, Italy, France, Portugal, Ireland, Finland, Slovenia, Poland, Netherlands, Turkey.

The primary contributions to this paper were made by the first three authors, while the fourth and fifth authors served as industrial problem owners with domain knowledge.

References

[1] M. Ahmed, A. N. Mahmood, and J. Hu. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, 60:19–31, 2016.

[2] A. A. Alli and M. M. Alam. The fog cloud of things: A survey on concepts, architecture, standards, tools, and applications. *Internet of Things*, 9:100177, 2020.

[3] U. Aslam, E. Batool, S. N. Ahsan, and A. Sultan. Hybrid network intrusion detection system using machine learning classification and rule based learning system. *International Journal of Grid and Distributed Computing*, 10(2):51–62, 2017.

[4] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

[5] Z. Chen, N. Lv, P. Liu, Y. Fang, K. Chen, and W. Pan. Intrusion detection for wireless edge networks based on federated learning. *IEEE Access*, 8:217463–217472, 2020.

[6] T. T. Dang, H. Y. Ngan, and W. Liu. Distance-based k-nearest neighbors outlier detection method in large-scale traffic data. In *2015 IEEE International Conference on Digital Signal Processing (DSP)*, pages 507–510. IEEE, 2015.

[7] A. Dehlaghi-Ghadim, A. Balador, M. H. Moghadam, and H. Hansson. Anomaly detection dataset for industrial control systems. *(in press)*, 2023.

[8] A. Dehlaghi-Ghadim, A. Balador, M. H. Moghadam, H. Hansson, and M. Conti. ICSSIM—a framework for building industrial control systems security testbeds. *Computers in Industry*, 148:103906, 2023.

[9] F. Firouzi, B. Farahani, and A. Marinšek. The convergence and interplay of edge, fog, and cloud in the ai-driven internet of things (iot). *Information Systems*, 107:101840, 2022.

[10] J. L. Guerra, C. Catania, and E. Veas. Datasets are not enough: challenges in labeling network traffic. *Computers & Security*, page 102810, 2022.

[11] E. Hemberg, J. R. Zipkin, R. W. Skowyra, N. Wagner, and U.-M. O'Reilly. Adversarial co-evolution of attack and defense in a segmented computer network environment. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1648–1655, 2018.

[12] P. Kairouz, H. B. McMahan, and et al. Advances and open problems in federated learning, 2021.

[13] R. Kale, Z. Lu, K. W. Fok, and V. L. Thing. A hybrid deep learning anomaly detection framework for intrusion detection. In *2022 IEEE 8th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*, pages 137–142. IEEE, 2022.

[14] D. T. Larose and C. D. Larose. *Discovering knowledge in data: an introduction to data mining*, volume 4. John Wiley & Sons, 2014.

[15] M. Lavassani, J. Åkerberg, and M. Björkman. From brown-field to future industrial networks, a case study. *Applied Sciences*, 11(7):3231, 2021.

[16] A. Lemay and J. M. Fernandez. Providing {SCADA} network data sets for intrusion detection research. In *9th Workshop on Cyber Security Experimentation and Test (CSET 16)*, 2016.

[17] Q. Li, Z. Wen, Z. Wu, S. Hu, N. Wang, Y. Li, X. Liu, and B. He. A survey on federated learning systems: Vision, hype and reality for data privacy and protection. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2021.

[18] Y. Luo, Y. Xiao, L. Cheng, G. Peng, and D. Yao. Deep learning-based anomaly detection in cyber-physical systems: Progress and opportunities. *ACM Computing Surveys (CSUR)*, 54(5):1–36, 2021.

[19] T. Markovic, M. Leon, D. Buffoni, and S. Punnekkat. Random forest based on federated learning for intrusion detection. In *IFIP International Conference on Artificial Intelligence Applications and Innovations*, pages 132–144. Springer, 2022.

[20] T. M. Mitchell. Machine learning, 1997.

[21] S. Moore. Gartner predicts by 2025 cyber attackers will have weaponized operational technology environments to successfully harm or kill humans. *Gartner Inc, MA, US, July*, 21, 2021.

[22] 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*, 9(4):2545–2554, 2021.

[23] S. S. Noureen, S. B. Bayne, E. Shaffer, D. Porschet, and M. Berman. Anomaly detection in cyber-physical system using logistic regression analysis. In *2019 IEEE Texas Power and Energy Conference (TPEC)*, pages 1–6. IEEE, 2019.

[24] G. Pang. The ai chip race. *IEEE Intelligent Systems*, 37(2):111–112, 2022.

[25] Ponemon Institite. Data Center Downtime at the Core and the Edge: A Survey of Frequency, Duration and Attitudes. Technical report, LLC, 2021.

[26] S. Raschka. Model evaluation, model selection, and algorithm selection in machine learning. *arXiv preprint arXiv:1811.12808*, 2018.

[27] N. Reichenberg. Improving security via proper network segmentation. *Security week*, 20, 2014.

[28] P. A. A. Resende and A. C. Drummond. A survey of random forest based methods for intrusion detection systems. *ACM Computing Surveys (CSUR)*, 51(3):1–36, 2018.

[29] N. K. Sahu and I. Mukherjee. Machine learning based anomaly detection for iot network:(anomaly detection in iot network). In *2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184)*, pages 787–794. IEEE, 2020.

[30] P. E. Strandberg. *Automated System-Level Software Testing of Industrial Networked Embedded Systems*. PhD thesis, Mälardalen University, 2021.

[31] P. E. Strandberg, D. Söderman, A. Dehlaghi-Ghadim, M. Leon Ortiz, T. Markovic, S. Punnekkat, M. Helali Moghadam, and D. Buffoni. The Westermo network traffic data set, 2023. Available online at: `https://github.com/westermo/network-traffic-dataset`, last accessed on 2023-04-12.

[32] Z. Tang, H. Hu, and C. Xu. A federated learning method for network intrusion detection. *Concurrency and Computation: Practice and Experience*, 34(10):e6812, 2022.

[33] M. A. Umer, K. N. Junejo, M. T. Jilani, and A. P. Mathur. Machine learning for intrusion detection in industrial control systems: Applications, challenges, and recommendations. *International Journal of Critical Infrastructure Protection*, page 100516, 2022.

[34] M. Zambon, R. Lawrence, A. Bunn, and S. Powell. Effect of alternative splitting rules on image processing using classification tree analysis. *Photogrammetric Engineering & Remote Sensing*, 72(1):25–30, 2006.

[35] J. Åkerberg, J. Furunäs Åkesson, J. Gade, M. Vahabi, M. Björkman, M. Lavassani, R. Nandkumar Gore, T. Lindh, and X. Jiang. Future industrial networks in process automation: Goals, challenges, and future directions. *Applied Sciences*, 11(8):3345, 2021.