Journal of Systemics, Cybernetics and Informatics (2023) 21(3), 29-36
https://doi.org/10.54808/JSCI.21.03.29

Using Federated Learning for Collaborative Intrusion Detection Systems

Matteo RIZZATO
Advestis part of Mazars, Tour Exaltis, 61 rue Henri Regnault
Courbevoie, 92400, France

Youssef LAAROUCHI
EDF R&D, Lab Paris-Saclay, 7 boulevard Gaspard Monge
Palaiseau, 91120, France

Christophe GEISSLER
Advestis part of Mazars, Tour Exaltis, 61 rue Henri Regnault
Courbevoie, 92400, France

ABSTRACT [1]

Neural networks have become cutting edge machine learning models for detecting network attacks. Traditional implementations provide fast and accurate predictions, but require centralised storage of labelled historical data for training. This solution is not always suitable for real-world applications, where regulatory constraints and privacy concerns hamper the collection of sensitive data into a single server. Federated Learning has recently been proposed as a framework for training a centralised model without the need to share data between different providers. We use the CICIDS2017 dataset provided by the Canadian Institute of Cybersecurity to demonstrate the benefits of Neural Networks-based Federated Learning for the detection of the most relevant types of network attacks. We conclude that a federated-trained neural network outperforms locally-trained models (at iso-architecture) in terms of F1-score and False Negative detection ratio. Further, such model has a minor loss of performance and convergence rapidity compared to a model trained over a hypothetical centralised dataset.

Keywords: Federated Learning · Neural Networks · Deep Learning · Cybersecurity · Multi-class classification.

1 INTRODUCTION

Computer attacks are becoming increasingly concerning in the current technological landscape, affecting both the availability (e.g. DoS attacks) and the quality of the delivered services (e.g. attacks on the integrity of machines, ransomware and document disclosure threats). It is therefore crucial to take into account computer attacks during the design and the operational stage of the system. In this context we focus on intrusion detection methods. In the past few years there has been growth of IDS (Intrusion Detection System) approaches and derivations based on network protocol analysis (NIDS) or on host logs (HIDS). These new methods all converge on the massive use of analysed data [8], naturally leading to the deployment of artificial intelligence algorithms to improve the performance of analyses [10]. However, data are not always available due to their sensitive nature and strict confidentiality rules. Therefore, the analyses for IDS can only be processed locally. We put ourselves in the context of a data collections hosted in different (and potentially competing) organisations. Despite this competitive environment, organisations are individually interested in having a better IDS that takes into account the state of the threat as a whole. In this paper, we present the effects of removal of this lock by proposing an approach based on Federated Learning (FL) [3, 7, 11]. FL allows to optimise a machine learning algorithm over a decentralised dataset without sharing the data across the clients (users) that own them. The resulting model is then available for inference to all the clients and no information on the original data can be retrieved [1, 4, 5]. We show in a first step that FL-based IDS are as efficient as classical IDS. Secondly, we show that the FL approach allows to base the prediction on a larger volume of data, and consequently, to improve the local performance of each organisation.

2 DATASET DESCRIPTION

In order to evaluate the performance of FL applied to network attack detection we choose the Intrusion Detection.

[1] We thank Nader Narcisse for peer-editing our manuscript.

Evaluation Dataset[2] (CIC-IDS2017) provided by the Canadian Institute of Cybersecurity[3] [9]. These data are made available to the scientific community for research on data mining for cyber purposes. The architecture on which the capture of network frames was based is quite classical and depicted in Fig. 1: a victim network of about 10 workstations with different operating systems connected to 2 routers. The external interface router is attacked by a network of machines belonging to an external attacker. The attack frames are then captured by the capture server which provides us with the final dataset.

Machine Learning dataset

Raw dataset The raw dataset includes the captures network traffic and system logs of each machine as gathered by the capture server. The data cover a period from 9 a.m., Monday, July 3, 2017 until 5 p.m. on Friday July 7, 2017, for a total of 5 days. The raw information can be converted into tabular labelled data via a feature extractor - a processed dataset is publicly available[4] and used for this work, the extractor deployed being CICFlowMeter[5].

Processed dataset for Machine Learning The resulting dataset is composed of 2,544,042 data points, each being described by the 83 unique features, plus the label specifying the type of traffic it refers to. As for the labels, 15 types of traffic are identified: Benign, Bot, DDoS, DoS GoldenEye, DoS Hulk, DoS Slowhttptest, DoS slowloris, FTP-Patator, Heartbleed, Infiltration, PortScan, SSH-Patator, Web Attack-Brute Force, Web Attack-Sql Injection, Web Attack-XSS. The partition of data points into the different classes is detailed in Tab. 1. We gather them in $n_c = 7$ classes: Benign, Bot, Brute Force, DOs/DDOS, Heartbleed, Infiltration, PortScan. We can drop 11 features as they do not present any variance across the dataset, hence being uninformative: Bwd Avg Packets/Bulk, Bwd Avg Bytes/Bulk, Fwd Avg Bulk Rate, Fwd Avg Packets/Bulk, Fwd Avg Bytes/Bulk, CWE Flag Count, Fwd URG Flags, Bwd URG Flags, Bwd PSH Flags, CWE Flag Count, Bwd Avg Bulk Rate. We finally remove the features providing the traffic ID and we feed the algorithm described in Sec. 4 with a cleaned dataset (CDS) including the remaining 65 features.

Table 1: Number of samples per class and relative percentage. Percentages are rounded at the second digit.

Benign	599945	58%
Bot	1944	0.19%
DDoS	128014	12%
DoS GoldenEye	10286	1.0%
DoS Hulk	172846	17%
DoS Slowhttptest	5228	0.58%
DoS slowloris	5385	0.52%
FTP-Patator	5931	0.58%
Heartbleed	11	0.0011%
Infiltration	36	0.0035%
PortScan	90694	8.8%
SSH-Patator	3219	0.31%
Web Attack-Brute Force	1470	0.14%%
Web Attack-Sql Injection	21	0.002%
Web Attack-XSS	652	0.063%

3 EXPERIMENT DESIGN

Training set

Even when a centralised dataset is available, network attack detection is a challenging task per se due to the strong level of asymmetry across different classes: the benign traffic points are over-represented ($\sim 60\%$ for the entire dataset) while most types of attack account for less than 1%. Commonly, we can deal with this problem by re-sampling the minority classes hence balancing the data points distribution.

In the context of FL, training set distributions present new key properties that a federated optimisation procedure must be tested against. First of all, the clients' dataset may itself not be balanced. Furthermore, a decentralised dataset is typically not independent nor identically distributed (non-IID), i.e. each client has different data distributions and has a biased sub-sample of the virtual centralised one. It is important to account for these properties when evaluating the robustness of a federated training. For this reason, starting from the 32 most populated machines[6] in the dataset described CDS in Sec. 1, we set up two different decentralised training sets (DS1 and DS2):

[2] `https://www.unb.ca/cic/datasets/ids-2017.html`
[3] `https://www.unb.ca/cic/`
[4] `http://205.174.165.80/CICDataset/CIC-IDS-2017/Dataset/`
[5] `http://ww12.netflowmeter.ca/`
[6] As explained later in the section, user model training is not performed in parallel, implying a user-time per round proportional to the number of machines involved in the collaborative learning. We picked a limited number of machines to mitigate the time per round, while ensuring all the classes are represented in the original mutual relative proportions. The *Destination IP* available as feature in the dataset is used to identify these machines.

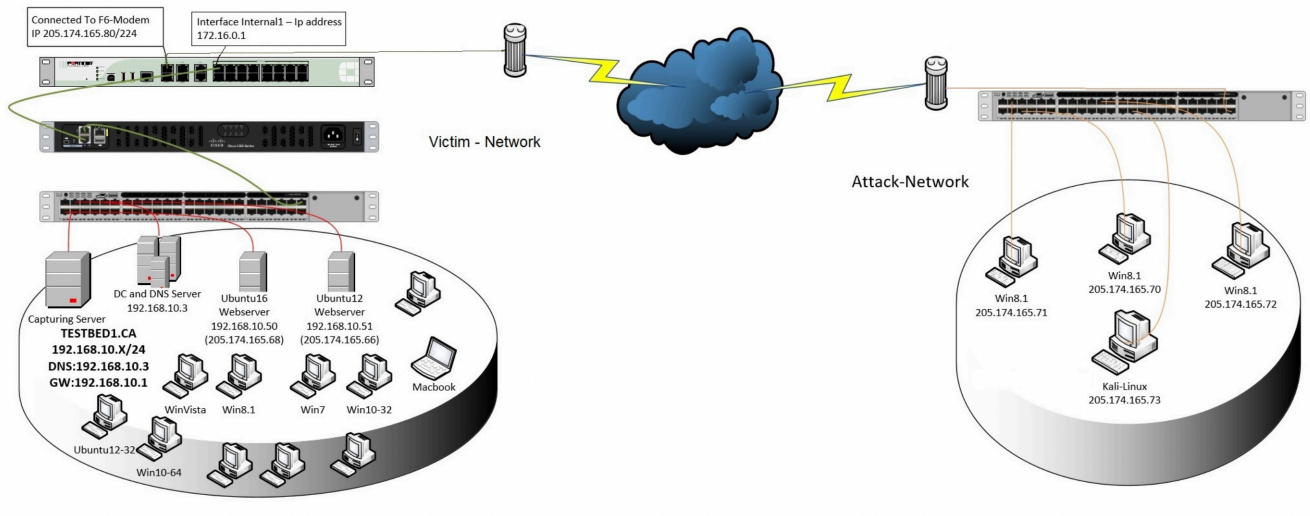

Fig. 1: Global architecture for data Acquisition. Original figure from [9]

we describe the procedure to obtain them in Tab. 2. The training/test split is set to 70% and such split is done at machine level. Further, we build two possible centralised datasets (B1 and B2) to be deployed in a classical centralised training environment. While it is not possible in real-life applications to centralise remote private datasets, in this simulation we build two possible benchmarks to evaluate the impact of FL on the validation performance.

Validation metrics and test set

In Sec. 4 we thoroughly compare the performance of the federated training against the two benchmarks described in Tab. 2. Within a classical Deep Learning framework the convergence rapidity is usually measured in terms of the number of epochs[7]. In FL it may be tricky to identify a comparable quantity due to the co-existence of several local models being trained altogether. Following [2], we utilise as unit of training time a single weight update in respectively the centralised architecture (*batch*) and the central server (*round*) for the benchmarks and the collaborative optimisation. We call this algorithmic unit step s. Please note that we do not account for any delays that may occur in real-life application related to communication issues. Accounting for these effects is beyond the scope of this work which is meant to be a controlled study in a simulation environment. Furthermore, due to the availability of a single Graphics Processing Unit (GPU)[8], a round of federated training entails a sequential update of the decentralised models, hence increasing the user-time per round. In real life applications, each decentralised model can rely on its own computational resources allowing parallel training of the secondary models. Therefore, using the actual user-time as time variable for our simulations would penalise federated trainings compared to classical ones and would not account for unavoidable latencies in real networks.

In terms of performance metrics, we plot the F_1 score in Eq. 1 and the fraction of false negative detections fn as functions of the training unit s. These metrics are computed from the precision p and the recall r which are obtained by aggregating the total number of true positive detections n_{tp}, false positive detections n_{fp} and false negative detections n_{fn} across all the data points observed at step s.

$$\mathrm{F}_1 = 2\frac{p\,r}{p+r}, \quad p = \frac{n_{\mathrm{tp}}}{n_{\mathrm{tp}}+n_{\mathrm{fp}}}, \; r = \frac{n_{\mathrm{tp}}}{n_{\mathrm{tp}}+n_{\mathrm{fn}}}. \quad (1)$$

The test set considered for the model evaluation (the central server in case of a federated training) is obtained by centralising the single-machine test sets with no re-sampling being performed. Federated and centralised trainings are both evaluated against the same dataset via the same metrics at each step s.

4 FEDERATED LEARNING AND NEURAL NETWORK ARCHITECTURE

As it is possible to grasp from the latest reviews on the topic [3, 7, 11], FL has received significant interest recently, both from research and applied perspectives.

[7] Depending on the numerical implementation, one epoch may or may not correspond to a single model weight update, or step (1 batch of data). In line with `TensorFlow` nomenclature, we assume that one epoch is defined by a given number of steps.

[8] NVIDIA Tesla T4, 15109MiB of virtual memory.

Table 2: Procedure to obtain the training sets for the different experiments proposed in our study. For the decentralised dataset, actions are applied on a randomly sampled fraction of the data for each client. The train/test split is 70%. The data not retained for training are centralised and deployed as a test set for both the training frameworks.

Decentralised datasets

Id.	Statistical properties		Description
DS1	Non-IID	Balanced	Destination IP in CDS is used to assign data points to decentralised users. The 32 most populated users are kept. Classes in clients are balanced via re-sampling. Final dataset size is unchanged.
DS2	Non-IID	Non-balanced	Destination IP in CDS is used to assign data points to decentralised users. The 32 most populated users are kept.

Benchmarks

Id.	Statistical properties	Description
B1	Balanced	32 most populated machines in CDS are centralised. Classes are balanced via re-sampling.
B2	Non-balanced	32 most populated machines in CDS are centralised.

This is due to it providing an optimisation framework for machine learning algorithms capable of bypassing important data-related challenges such as privacy or the cost of centralising databases placed over different servers. Within the realm of neural networks, different optimisation algorithms have been proposed [2, 6, 12]. For this work, we deploy the widely used `FederatedAveraging` algorithm [2] as it is available in the FL high-level interfaces offered by the library `TensorFlow Federated`[9]. In Tab. (4) we present the architecture we retained for our study.

`FederatedAveraging` algorithm

The `FederatedAveraging` algorithm was first introduced in [2] and it is summarised in Algorithm 1. The main idea is that an instance of all the weights and biases ω of the central model are present on the clients. Upon availability of the client k, during a round t (step) of training it receives the weights/biases from the central model and performs a local update of its model instance ω_k against the training dataset $\mathcal{P}_k$ of size n_k available locally. The client optimiser looks at data batches of size B for E epochs of training. Once the decentralised models have been updated, their weights $\{\omega_k\}_{k=1,\dots,S_k}$ are sent back to the central server which aggregates them via a weighted average accounting for the respective number of local samples. Given this framework, the hyper-parameters retained for our experiences are listed in Tab. 6 for both the centralised trainings serving as benchmarks and the collaborative ones. At the best of our knowledge, `TensorFlow Federated` high-level interfaces do not allow to fine tune the clients' hyper-parameters. During our study we observed a drift in the central

[9] `https://www.tensorflow.org/federated`

Algorithm 1 The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:
  initialize w_0
  for each round $t = 1, 2, \dots$ **do**
    $m \leftarrow \max(C \cdot K, 1)$
    $S_t \leftarrow$ (random set of m clients)
    for each client $k \in S_t$ **in parallel do**
      $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
    end for
    $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$
  end for

ClientUpdate(k, w): *// Run on client k*
  $\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
  for each local epoch i from 1 to E **do**
    for batch $b \in \mathcal{B}$ **do**
      $w \leftarrow w - \eta \nabla \ell(w; b)$
    end for
  end for
  return w to server

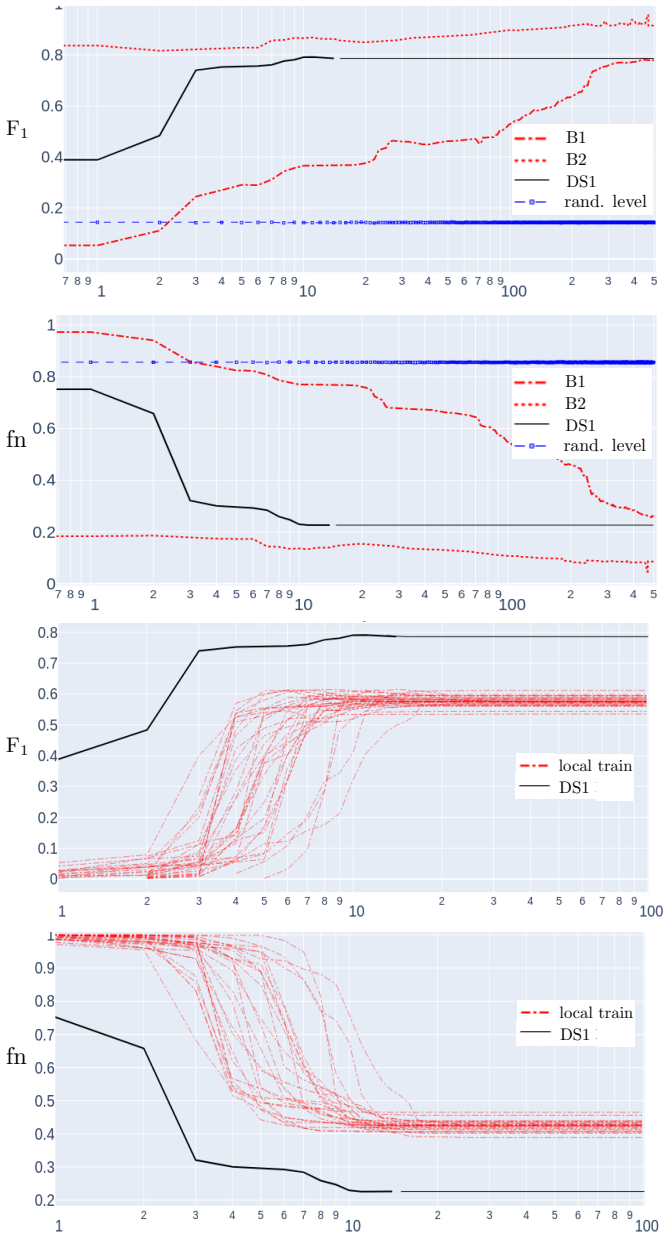

Fig. 2: Performance analyses of a federated training over the dataset DS1. From the top: *a)* F_1 score obtained for DS1 (black line) against centralised training for B1 (red dash-dotted line), centralised training for B2 (red dotted line) and random prediction (dashed blue line with box-shaped markers). The federated training has been interrupted at the best performing point during training and this point was extended for visualisation purpose; *b)* same analysis as in the first panel, but for the false negative detection ratio fn; *c)* F_1 score obtained for DS1 (black line) against locally-trained machines in the 32 clients' databases retained for this study; *d)* same analysis as in the third panel, but for the false negative detection ratio fn.

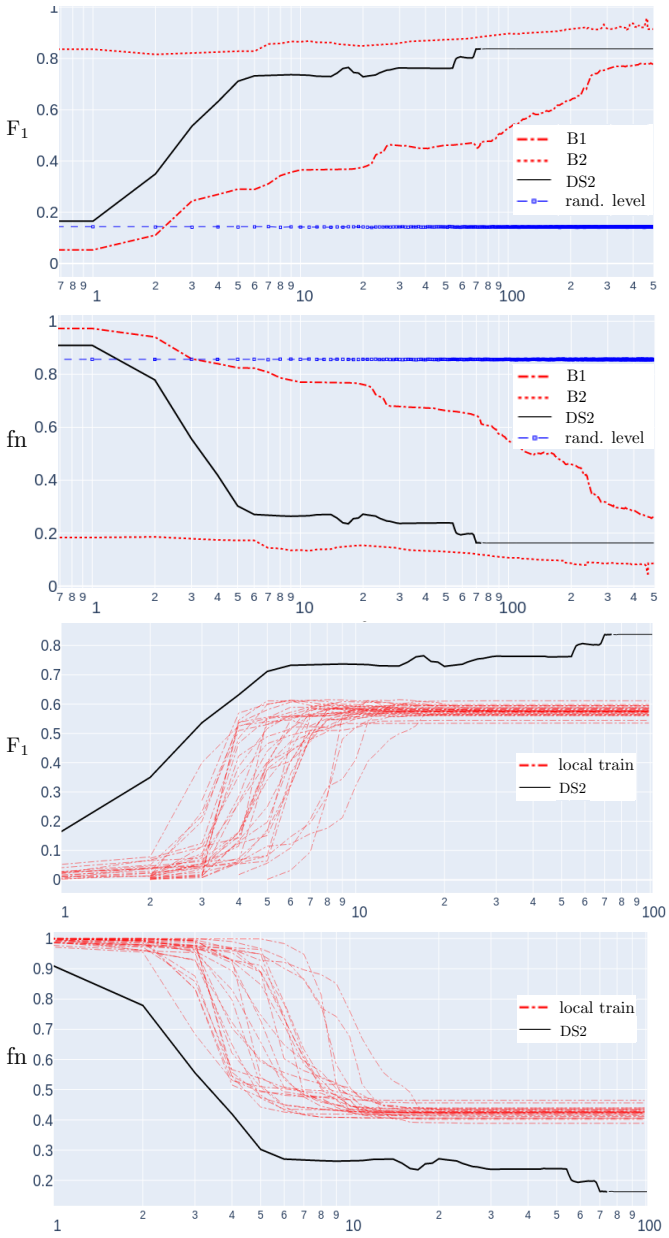

Fig. 3: Performance analyses of a federated training over the dataset DS2. From the top: *a)* F_1 score obtained for DS2 (black line), centralised training for B1 (red dash-dotted line), centralised training for B2 (red dotted line) and random prediction (dashed blue line with box-shaped markers). The federated training has been interrupted at the best performing point during training and this point was extended for visualisation purpose; *b)* same analysis as in the first panel, but for the false negative detection ratio fn; *c)* F_1 score obtained for DS2 (black line) against locally-trained machines in the 32 clients' databases retained for this study; *d)* same analysis as in the third panel, but for the false negative detection ratio fn.

Table 3: Maximum performance obtained by FL over the two datasets retained as function of the fraction of clients being available for the round of server update. Maximum performance is defined by the minimum ratio of false negative detection.

DS1

Frac. client	0.1	0.3	0.5	1.0
Min. fn	**0.24**	**0.20**	**0.22**	**0.22**
Round	4	4	3	15
$\mathbf{F}_1$	0.78	0.80	0.78	0.78

DS2

Frac. client	0.1	0.3	0.5	1.0
Min. fn	**0.41**	**0.24**	**0.20**	**0.16**
Round	24	72	12	80
$\mathbf{F}_1$	0.58	0.75	0.80	0.84

server performance while training for several rounds. To prevent this effect we made use of early-stopping. This pheonomen is probably due to some clients overfitting their own training set hence sending back to the global model strongly biased weights due to the non-IID nature of the data. As the focus of this work is on the benefits of Federated Learning for network attack detection and since the conclusions in Sec. 4 are not affected by this behaviour, we do not investigate this issue further and we defer it to future works.

5 EXPERIMENTAL RESULTS

We test the performance of the optimisation algorithm outlined in Sec. 4 for both the training set presented in Tab. 2 and for a varying fraction of clients: in real-life applications not all the users are available for communication at each round. After evaluating a federated decentralised benchmark where all 32 clients are always available, we test the robustness of the results by randomly choosing respectively 50%, 30%, and 10% of the available machines to communicate with the main server.

In the different panels in Fig. 2 we summarise the results for the training set DS1 when all clients are used for the server updates at each round. Starting from the top, we show the F_1 score and the false negative detection ratio as a function of the algorithmic unit s for the federated training and for the two benchmarks in Tab. 2. We also include the performance of a random prediction on the attack classes - such being the status of a novel client being installed with no historical training set available (cold start). Within the maximum number of epochs allowed, the benchmark B2 outperforms the early-stopped federated-trained neural networks: the F_1 score decreases by $\sim 15\%$ and the false negative detection ratio increases by a factor of ~ 2.5 when moving to a collaborative detection. As for the benchmark B1, FL implies no loss of performance at all with respect to the virtual centralised scenario. Furthermore, it reaches the maximum performance with fewer model updates. For real applications, if a new machine is installed with no historical data (cold start), retrieving the central model can ensure a F_1 score $\gtrsim 4.5$ higher and a false negative detection ratio ~ 4 times smaller than a random detection. Please note that the benchmarks B1 and B2 as defined in Sec. 1 provide a fictitious conservative landmark to compare with the federated neural networks performance. By definition, FL is the machine learning optimisation solution when data cannot be centralised. A much more relevant test is proposed in the last two panels of Fig. 2 where we compare the validation performance of federated training the model against locally training the same architecture over each of the 32 decentralised datasets. The test is kept unchanged: it includes all the test data coming from the entire population. The local models have been optimised via a constant learning rate $\eta = 10^{-3}$. The results are clear: FL leads to a major benefit with a gain of $\sim 27\%$ in terms of F_1 score when compared to the best performing machine. The ratio of false negative detection is instead reduced by a factor of ~ 1.7.

In Fig. 3 we perform the same analysis for the dataset DS2. While the performance figures overall have minor differences compared to those in the previous paragraph, the key results are unchanged. Compared to a cold start, a client benefiting from a collaborative learning can increase its F_1 score by a factor of ~ 5 and reduce the false negative detection ratio by a factor of ~ 5.3. Federated training a neural network extensively outperforms even the best performing locally-trained machine by $\sim 36\%$ in terms of the F_1 score while decreasing the fraction of false negative detection by a factor of ~ 2.3.

We conclude our analysis by testing the robustness of the results above when gradually reducing the number of clients available at each round. In Tab. 3, for the dataset DS1 it is possible to observe a negligible degradation of the performance even when reaching a minimal fraction (10%) of the clients joining the rounds of training. A loss of performance can be seen for the dataset DS2 when using only the 10% of the clients per round. The benefits of the collaborative training are still visible up to a fraction of clients of 0.3 and even in the extreme case above, the federated trained neural network performs no worse than the best performing locally-trained single machine.

6 CONCLUSIONS

In this paper we explored the potential benefits of Federated Learning applied to neural networks

for the collaborative detection of systems intrusion in terms of false negative detection ratios and F_1 score. The chosen optimisation algorithm is `FederatedAveraging` and the data used come from the publicly available CICIDS2017 dataset provided by the Canadian Institute of Cybersecurity. We first compared the Federated Learning approach with a virtual model trained on 2 possible centralised views of the peripheral dataset and with a random classification. When a balanced centralised dataset is used as benchmark, Federated Learning does not show any loss of performance and plateaus in much fewer steps. Furthermore, only by inheriting the weights from the central model, a newly installed machine (cold start) can see a reduction of its false negative detection ratio by a factor of ~ 4 to ~ 5 compared with a random classification. A second crucial result of the current study is that Federated Learning can boost the performance of intrusion detections when compared to locally trained models. For both the decentralised training dataset considered, the number of false negative detection is divided by a factor of ~ 2 (~ 1.7 for DS1 and ~ 2 for DS2) and the F_1 score is increased by $\sim 30\%$ ($\sim 27\%$ for DS1 and $\sim 36\%$ for DS2). Such performance is robust against a reduction of the fraction of clients participating in the rounds of training: a loss in performance is visible only below 30% users per round being involved in training for DS2, while no remarkable loss of performance is present for DS1.

7 ACKNOWLEDGMENT

We thank George Luta and Hans-Jörg Von Mettenheim for their critical review of the manuscript.

8 REFERENCES

[1] Augenstein, S., McMahan, H. B., Ramage, D., Ramaswamy, S., Kairouz, P., Chen, M., Mathews, R., and Aguera y Arcas, B. **Generative Models for Effective ML on Private, Decentralized Datasets**. *arXiv e-prints* (Nov. 2019), arXiv:1911.06679.

[2] Brendan McMahan, H., Moore, E., Ramage, D., Hampson, S., and Agüera y Arcas, B. **Communication-Efficient Learning of Deep Networks from Decentralized Data**. *arXiv e-prints* (Feb. 2016), arXiv:1602.05629.

[3] Brisimi, T. S., Chen, R., Mela, T., Olshevsky, A., Paschalidis, I. C., and Shi, W. **Federated learning of predictive models from federated Electronic Health Records**. *International Journal of Medical Informatics 112* (2018), 59–67.

[4] Dwork, C. **A Firm Foundation for Private Data Analysis**. *Commun. ACM 54*, 1 (jan 2011), 86–95.

[5] Dwork, C., McSherry, F., Nissim, K., and Smith, A. **Calibrating Noise to Sensitivity in Private Data Analysis**. In *Theory of Cryptography* (01 2006), vol. Vol. 3876, pp. 265–284.

[6] Konečný, J., Brendan McMahan, H., Ramage, D., and Richtárik, P. **Federated Optimization: Distributed Machine Learning for On-Device Intelligence**. *arXiv e-prints* (Oct. 2016), arXiv:1610.02527.

[7] Li, L., Fan, Y., Tse, M., and Lin, K.-Y. **A review of applications in federated learning**. *Computers & Industrial Engineering 149* (2020), 106854.

[8] Seng, S., Garcia, J., and Laarouchi, Y. **Implementation of a Stateful Network Protocol Intrusion Detection Systems**. *The International Conference on Security and Cryptography SECRYPT* (July 2022).

[9] Sharafaldin, I., Habibi Lashkari, A., and Ghorbani, A. **Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization**. pp. 108–116.

[10] Volkov, S. S., and Kurochkin, I. I. **Network attacks classification using Long Short-term memory based neural networks in Software-Defined Networks**. *Procedia Computer Science 178* (2020), 394–403. 9th International Young Scientists Conference in Computational Science, YSC2020, 05-12 September 2020.

[11] Yang, Q., Liu, Y., Chen, T., and Tong, Y. **Federated Machine Learning: Concept and Applications**. *arXiv e-prints* (Feb. 2019), arXiv:1902.04885.

[12] Zhu, H., and Jin, Y. **Multi-objective Evolutionary Federated Learning**. *arXiv e-prints* (Dec. 2018), arXiv:1812.07478.

APPENDICES

A ARCHITECTURE

Table 4: n_b batch size.

Layer type	Output shape	Number of parameters	Info.
Input	(n_b, 66)	0	
Dense	(n_b, 128)	8576	
Dropout	(n_b, 128)	0	rate: 0.1
Batch Norm.	(n_b, 18)	512	
LeakyRelu	(n_b, 128)	0	neg. slope: 0.2
Dense	(n_b, 64)	8256	
Dropout	(n_b, 64)	0	rate: 0.1
Batch Norm.	(n_b, 64)	256	
LeakyRelu	(n_b, 64)	0	neg. slope: 0.2
Dense	(n_b, 64)	4160	
Batch Norm.	(n_b, 64)	256	
LeakyRelu	(n_b, 64)	0	neg. slope: 0.2
Dense	(n_b, 7)	455	
Sofrmax	(n_b, 7)	0	

B HYPER-PARAMETERS

Table 5: Hyper-parameters retained for the different training sets listed in Tab. 2. When a hyper-parameter is set to vary during training, the notation $A \xrightarrow{c} B$ indicates that the value has been changed from A to B at step $s = c$.

Benchmark B1

Optimiser	Adam $\eta = \{0.01 \xrightarrow{10} 0.001 \xrightarrow{100} 0.0003 \xrightarrow{280} 0.0001\}$
Batch size	2048
Epochs Max.	500
Steps/Epoch	1

Benchmark B2

Optimiser	Adam $\eta = \{0.03 \xrightarrow{12} 0.01 \xrightarrow{22} 0.003\}$
Batch size	264
Epochs	500
Steps/Epoch	1

Table 6: Hyper-parameters retained for the different training sets listed in Tab. 2. When a hyper-parameter is set to vary during training, the notation $A \xrightarrow{c} B$ indicates that the value has been changed from A to B at step $s = c$.

Federated training DS1

Server Optimiser	Adam $\eta = \{0.03 \xrightarrow{5} 0.0003\}$
Client Optimiser	Adam $\eta = \{0.03 \xrightarrow{5} 0.0003\}$
Client batch size	$B = \{32 \xrightarrow{5} 12\}$
Client n. epochs	$E = \{50 \xrightarrow{4} 20\}$
Rounds Max.	500
Fraction clients per round	1.0, 0.5, 0.3, 0.1

Federated training DS2

Server Optimiser	Adam $\eta = \{0.03 \xrightarrow{12} 0.01 \xrightarrow{22} 0.003 \xrightarrow{35} 0.001 \xrightarrow{70} 0.0003\}$
Client Optimiser	Adam $\eta = \{0.03 \xrightarrow{12} 0.01 \xrightarrow{22} 0.003 \xrightarrow{35} 0.001 \xrightarrow{70} 0.0003\}$
Client batch size	$B = \{32 \xrightarrow{80} 16\}$
Client n. epochs	$E = \{50 \xrightarrow{14} 20 \xrightarrow{21} 10 \xrightarrow{27} 5 \xrightarrow{45} 1\}$
Rounds Max.	500
Fraction clients per round	1.0, 0.5, 0.3, 0.1