Federated Learning based Approach for Intrusion Detection in Imbalanced and Non-IID Setting

1st Md. Mohaiminul Islam
dept. of Computer Science and Engineering
Bangladesh University of Engineering and Technology (BUET)
Dhaka, Bangladesh
mohaiminmahim@gmail.com

***Abstract*—Intrusion detection Systems(IDS) can identify both known and unknown attacks from properties of network traffic and has been an essential part of network security. Tradition Deep learning and Machine learning models have been widely accepted as the norm to develop anomaly based intrusion detection schemes for a while. Federated learning is a relatively new and well established approach to train machine learning models in a highly distributed setting. Machine learning models that incorporate federated architecture are inherently superior in utilizing available resources efficiently and upholding user data privacy. In our work we show federated approach fused with traditional machine learning can benefit Intrusion Detection Systems over real life scenarios. These situations include Non-identical distribution of attack data as well highly imbalanced ratio of existing benign traffic to malicious traffic data.**

***Index Terms*—Federated learning, intrusion detection, imbalanced, NSL-KDD**

I. Introduction

Distributed systems are at the core of many recent technological advances. Specially recent outburst of consumer based softwares and smart devices have led many large corporations with billions of users to adopt distributed architectures for their networking and data warehousing etc. But alongside these advances cyber threats have also persistently grown in scale. Often times attacker groups dismantle networks of millions of devices resulting is mass interruption of services or theft of critical user information etc. This is why detecting security breaches and taking fast preventive measures are an open challenge in advanced communication (5G and beyond 5G) protocols. At the same time preserving the confidentiality and integrity of users' data is another major area of concern. For detecting network intrusion detection machine learning models, specially deep learning models are widely accepted as go to for several years. The obvious problem which lies here is to train such traditional ML models we would be required to collect user data and store that data at a centralized location for training purposes. In [3], the authors discovered that a high packet loss rate causes the prediction accuracy to deteriorate as network size increases (the more switches, the less accurate the prediction). But collecting vast amount of network data is impractical to begin with in 5G and B5G environments. Moreover, such centralised collection of user data is always risky and might not be possible due to recent trends in data privacy standards. To rectify the above issue researcher at Google proposed a new approach [4], coining the term *Federated Learning*, which allows for collaborative learning amongst the devices without the requirement for data sharing with a central server. To put it another way, traditional ML/DL models can now be trained on end devices using it's local data in a decentralized manner which shares the learning parameters instead of the data itself [5], [6]. It is an iterative process that improves the model in each iteration(also known as communication rounds) by two steps - *a)localized training with local data and, b)sharing of model weights to central agent that an accumulate the learning benefits of all the localized agents*. When a communication round is initiated, the centralized agent selects a subset of clients to include in that iteration, while and disseminates to them its current global model [7]. when selected clients receive the global model, each client uses its own data for the local training. Then these clients send back their new model (i.e., the learned parameters) to the FL server for global aggregation (Fig. I) [9]. This process is repeated various rounds until the desired performance is achieved. To summarize, a federated learning approach consists of three main phases: local update, local training and global aggregation. The advocates of FL demonstrated that, each client can utilize the other clients' data(or the knowledge extracted from it) without ever knowing or handling their privacy-sensitive personal data. Even the central entity is devoid of any knowledge of the clients' own data [8]. Although FL provides a new dimension that can benefit many areas in applied machine learning, it's implementation in most areas, specially in intrusion detection still poses a number of real world challenges that needs to be addressed [9], [10].Some of these challenges encompass-

- How can Federated architectures reduce communication overhead and become feasible enough to be incorporated in real time applications [12], [14]?
- How to mitigate adversarial security threats(e.g. federated poisoning attacks) [17]?
- How to build a robust architecture that excels in a heterogeneous client environment [11]?
- Devising and deployment of federated solutions in low power computational devices. [15], [16]

Previous works regarding FL in intrusion detection scenario mostly failed to leverage realistic datasets in this context. Also other works that try to to tackle the open challenges is mostly limited to very generalized results using rudimentary datasets like the MNIST which is mainly used for image classification tasks [11]. Hence a comprehensive study that aims to alleviate the challenges posed using benchmark datasets is lacking in this domain. In this paper we aim to bridge that gap by experimenting and consolidate our results on the NSL-KDD benchmark dataset for network intrusion detection.

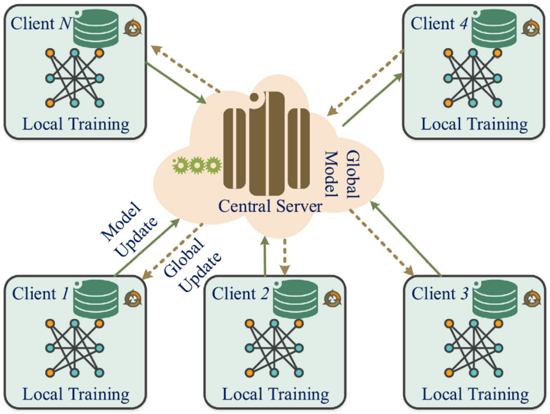


Fig. 1. General Federated Learning Architecture

II. Related work

Further exploration of Federated algorithms and their specification, the scenario envisioned by [4] describes the decentralized non-IID optimization issue in terms of Federated learning. They propose the *Federated Averaging(FedAVG.)* algorithm that now serves as a powerful baseline for many researches using FL. In this algorithm, several client nodes or edge devices use their personal data to collaboratively train a global model, dictated by the coordination of a central agent. The role of this agent is to average the parameters of the models sent by the clients and return the resulting global model to them. This learning method is continuously repeated until a terminating condition is met. FL has advanced a long way from there. Many surveys [18], [19] are done to review the latest advances in that field. Also a few works exist that paints an overall picture about the current state of federated learning based techniques in IDS systems [10].Because of its dependency on server client communication federate architectures can face many security threats. The work of [20] reviews some of the issues that can arise when an attacker or adversary is introduced, along with the best known counter measures till now. The work of [21] introduces two new aggregation functions that are robust and are based on the coordinate-wise median and the coordinate-wise trimmed mean of the models sent by the clients.In [22] the authors proposed resampling to reduce heterogeneity in the distribution of the models sent by the clients. It is meant to be applied before using a robust aggregation function, and it aims at reducing the side-effects that such a function has when applied to models trained with non-IID datasets. Zhao et al. [23] proposed that the server shares a small subset of public data with heterogeneous clients to reduce the weight divergence between their local models, hence increasing training stability and bolstering robust performance. Although this has the prerequisite of such public data being sufficiently available. The work of [24], solves the same issue by adding a regularisation term to the local optimization, Which limits the weight divergence between local and global models. Both approaches rsult in a single global model which aims to improve accuracy. However, under some non-iid setting (specially where participating clients have conflicting optimization goals), a single joint global model that is optimal for all clients is not possible [25]. Ghosh et al. [26] use K-means to cluster client updates for the purpose of identifying and isolating byzantine clusters of clients prior to the averaging step. Finally [11] develops a hierarchical federated learning method, that covers a wide range of non-iid settings, similarity measures and allows for training on only a fraction of clients per round of communication. They also employ a single clustering step reducing the load of the clustering procedure on the aggregation server.

III. Methodology

In this section we describe the architectural layout of our proposed model. It also entails the clients, their communication details and how we use the NSL-KDD Dataset for the training and validation of said model.

A. Clients

We create virtual clients in google colab to simulate a distributed server client architecture. All of these clients initialize a local model for training and validation. At the same time a global model is initialized. Then each client is provided with a different set of sample data. Though initialized identically with the same model the learning parameter of these local models diverge over several epochs depending on their own data. After a certain number of epochs all the clients halt their local training . In the next step, clients send only the learning parameters, i.e. weights to the central server where all the learning corresponding learning parameters are aggregated and the global model is updated with these parameters. A global validation report is generated and subsequently an update is sent to all of the participating clients with the aggregated parameters. The clients update their local model accordingly, which concludes a single phase of the whole training process known as a communication round. The local models are trained and aggregated for various such rounds.

B. Dataset and Preprocessing

In this study we used the NSL-KDD benchmark dataset, which has been used profusely in training intrusion detection systems both to recognize benign and malicious traffic. This dataset was developed to overcome the shortcomings of the original KDD'99 dataset. It contains data of 22 different variety of attacks as well as normal traffic. We execute all our experiments on NSL-KDD dataset, In table I the composition of the dataset is summarized.

TABLE I
SAMPLE DISTRIBUTION OF DIFFERENT ATTACK CLASSES

Category	Attack type	Train	Test
Dos	back	956	359
	land	18	7
	neptune	41214	4657
	pod	201	41
	smurf	2646	565
	teardrop	892	12
Probe	ipsweep	3599	141
	nmap	1493	73
	portsweep	2931	157
	satan	3633	735
R2L	ftp_write	8	3
	guess_password	53	1231
	imap	11	1
	multihop	7	18
	phf	4	2
	spy	2	0
	warezclient	890	0
	warezmaster	20	944
U2R	loadmodule	9	2
	buffer_overflow	30	20
	rootkit	10	13
	perl	3	2
Normal		67343	9711
Total		125973	18794

The data preprocessing steps are briefly explained below

- First we relabel the attack data into 5 generic classes, the reason for doing this is twofold. First of which is it increases model performance when we collapse similar subclass of attacks under one generic label. Secondly, it is easier to compare the results with preexisting work based on both the NSL-KDD and KDD '99 datasets.
- The *one-hot encoding* is used to process the dataset and convert symbolic features into numerical features. The protocol type, for instance, is the second attribute of the NSL-KDD data sample. TCP, UPD, and ICMP are the three options for the protocol type. The one-hot approach is converted into a computer-readable binary code, where TCP is represented by [1, 0, 0], UDP by [0, 1, 0], and ICMP by [0, 0, 1]. One-hot encoding was applied on the following features- *protocol type, service and flag*.
- The value of the original data may be too large, resulting in problems such as features with larger values dominating the smaller values, data processing overflows, and inconsistent weights so on. We use standard min-max scaler to normalize the continuous data into the range [0, 1]. Normalization processing eliminates the influence of the measurement unit on the model training, and makes the training result more dependent on the characteristics of the data itself. The formula is shown in equation 1.

$$r\prime = \frac{r - r_{min}}{r_{max} - r_{min}} \tag{1}$$

Where r stands for the numeric value of the given feature, r_{max} and r_{min} stands for the maximum and minimum possible values for that feature across the whole dataset and $r\prime$ stands for the normalized value.

IV. EXPERIMENTAL SETUP AND RESULTS

A. Model description

All the clients were initialized with a simple neural network with 2 hidden layers. The main goal of the experiments were to test the viability of federated architecture on an imbalanced benchmark dataset like NSL-KDD with both IID and Non-IID distribution of data. Hence we proceed with a very simple learning model and inquire whether it improves or not. This also should eliminate model specific results and produces more broader results applicable to any local or global model architecture. The detail and dimensions are of the layers are shown in table IV-A

Layer	Dimension	Activation Function
Input	-	-
First hidden layer	64	ReLu
Second hidden layer	64	ReLu
Third hidden layer	128	ReLu
Output layer	5	Softmax

We train the model for a predefined number of epochs and measure the following metrics. Stochastic gradient descent(Adam algorithm) was implemented that is based on adaptive estimation of first-order and second-order moments. According to Kingma et al.(2014) [1], the method is"computationally efficient, has little memory requirement, invariant to diagonal rescaling of gradients, and is well suited for problems that are large in terms of data/parameters".

1) **Categorical Cross Entropy:** This metric was calculated using the formula

$$\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \tag{2}$$

Where $M = 5$(the number of classes), y is a binary value indicating if class label c is the correct classification for observation o and p is the predicted probability observation o is of class c.

2) **Macro F1 score:** The $F1$ score can be interpreted as a harmonic mean of the precision and recall, which reaches its optimal value at 1 and worst score at 0. The relative weight of precision and recall to the $F1$ score are equal. The formula for the F1 score is:

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN} \tag{3}$$

3) **Macro F_beta score:** The weighted harmonic mean of recall and precision is derived as the F_beta score, which reaches its optimal value at 1 and its worst value at 0. The macro F_beta calculates this metric for each label, and find their unweighted mean, thus eliminating the imbalanced nature of the dataset. we can calculate the F_beta score with the following formula

$$F\_beta = (1 + \beta^2) \frac{Precision * Recall}{\beta^2 * Precision + Recall} \tag{4}$$

In other words, if $\boldsymbol{\beta < 1}$, the precision is $\boldsymbol{\beta}$ times more important than recall, and if $\boldsymbol{\beta > 1}$, it's the other way around. We set $\beta = 2.0$ for our experiments.

B. Experiment : Varying the number of Clients

The NSL-KDD dataset has seggregated training and testing data. At the start of training process both the training and testing datasets are randomly sub-divided into **n** data shards and distributed to the clients without loss of generality. So, the number of clients affects the quantity and distribution of data held by each client. Hence, it affects the learning process as a whole. To study the extent of this influence we initiated the learning with $n = 5, 10, 20$ clients and commensed this learning for 5 communication rounds, each consiting of 10 local epochs for all the clients. After each client has trained their local model with their local data for 10 epochs, their results are collected and weights are aggregated in the global server. This aggregation update is transmitted to all the clients who readjust their weights accordingly and start the learning for the next comm round. All the results for this section is summarized in Table II,III and IV

TABLE II
RESULTS FOR EXPERIMENT IV-B WHEN $n = 5$

Accuracy	Class	Precision	Recall	F1 Score
0.79	Normal	0.92	0.85	0.88
	DoS	0.75	0.91	0.82
	Probe	0.67	0.79	0.73
	U2R	0.86	0.20	0.33
	R2L	0.38	0.66	0.48
	Macro Avg.	0.72	0.68	0.65
	Micro Avg.	0.81	0.79	0.77

TABLE III
RESULTS FOR EXPERIMENT IV-B WHEN $n = 10$

Accuracy	Class	Precision	Recall	F1 Score
0.78	Normal	0.90	0.83	0.86
	DoS	0.71	0.91	0.80
	Probe	0.76	0.75	0.75
	U2R	0.86	0.17	0.29
	R2L	0.55	0.52	0.53
	Macro Avg.	0.75	0.64	0.65
	Micro Avg.	0.79	0.78	0.75

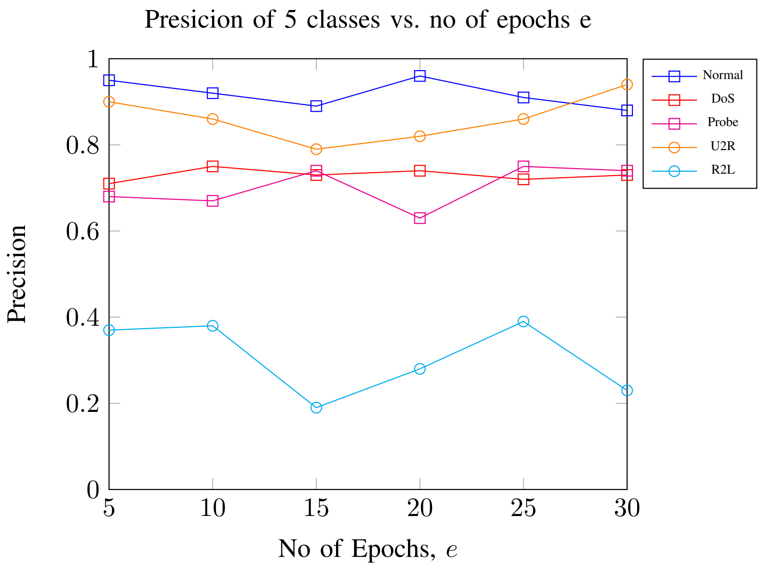
We can see from both overall model accuracy and F1 score of the minority classes, introducing more clients deteriorates the performance. This is due to the fact that already these classes are lacking in data. So, more clients result in overall very low amount of training data for each client for these classes. Hence, the performance downfall. This also proves when there is a lack of training data and no further data can be injected into the learning process federated learning performs best with a lower number of clients. That is why for the following experiments we set the number of clients $n = 5$.

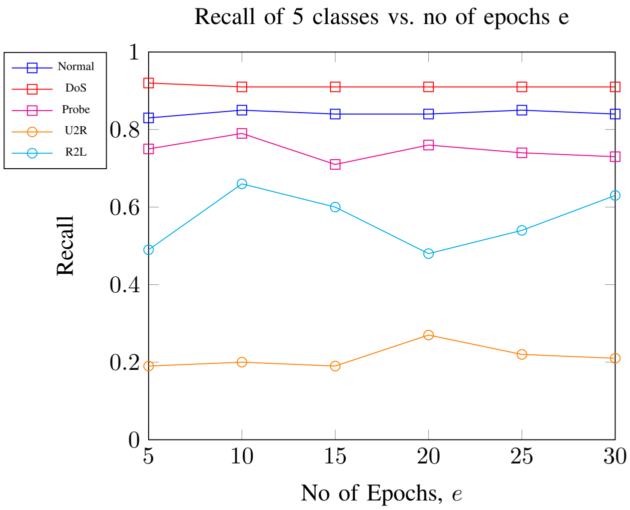
TABLE IV
RESULTS FOR EXPERIMENT IV-B WHEN $n = 20$

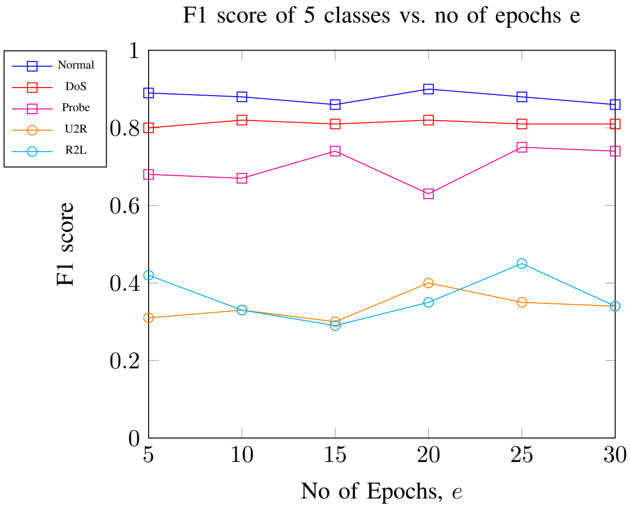
Accuracy	Class	Precision	Recall	F1 Score
0.77	Normal	0.90	0.84	0.87
	DoS	0.70	0.91	0.79
	Probe	0.75	0.72	0.74
	U2R	0.93	0.15	0.26
	R2L	0.42	0.46	0.44
	Macro Avg.	0.74	0.62	0.62
	Micro Avg.	0.80	0.77	0.75

C. Experiment : Varying the number of Epochs between Communication round

Generally in federated architecture each client communicates with the server after training their respective local models for a predefined number of epochs with their local data. Varying the amount of intercommunication round training is an obvious optimization problem because of the two main opposing factors at play. If the local training is done for a very short number of epochs, it can lead a node to send non-optimal weights to the server because the local model is unable to reach anywhere near convergence in such a short number of epochs. Gradually these non-optimal weights from each client will be aggregated, hence, the performance of the global model would deteriorate slowly . On the other hand, too many epochs of training between communication round creates an over reliance on local data and might even lead to overfitting of the local models. So, in the aggregation process if these locally biased parameters are communicated, it will lead to suboptimal performance of the global model. Thus, we conducted this experiment on the number of epochs between each communication round,e with $5, 10, 20, 25 and 30$ epochs. We set the number of clients $n = 5$ and the number of comm. round $c = 5$ as well. The following plots reflect the performance of the global model based on Precsion , Recall, F1 score and Accuracy







epochs	5	10	15	20	25	30
Accuracy	0.78	0.79	0.78	0.79	0.79	0.78

As hypothesized the global model performs best for all 5 classes if the training is continued for 15-25 epochs before each communication round. After that the model starts overfitting for that class. The difference is likely to due to the highly imbalanced nature of our dataset, as the classifier converges at different rates for each class.

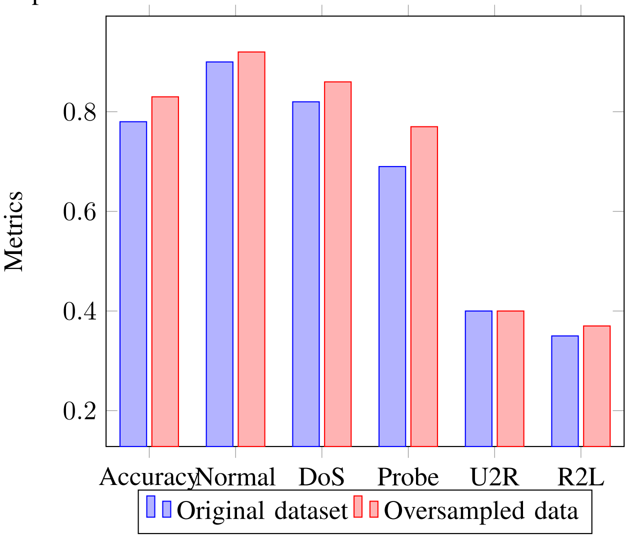
D. Experiment : Balancing the dataset

In this section we try to reconcile the imbalanced nature of NSL-KDD by creating several oversampling techniques. Mainly we attempt to balance out the classes by over sampling using Adaptive Synthetic (ADASYN) algorithm [2]. This algorithm is similar to SMOTE in nature but it generated different amount of synthetic samples depending on an estimated local distribution of the classes to be oversampled. The newly generated synthetic samples are close to the decision boundary to maximize the learning potential. Table V shows the relative abundance of all five classes after applying oversampling them.

TABLE V
DISTRIBUTION AFTER OVERSAMPLING

Class	Before Oversampling	After Oversampling
Normal	67343	67391
DoS	45927	67347
Probe	11656	67344
U2R	995	67343
R2L	52	67333

Figure IV-D presents the comparison between learning on original dataset vs the newly resampled dataset.The first set of bars show overall accuracy for both datasets and the rest compares the F1 score of each class both for original and resampled data.



If we analyze this we can see the overall accuracy has increased by nearly 5% . At the same time f1 score has better or nearly same for all the classes.

V. CONCLUSION

In this work we have shown federated learning approach can supersede the traditional ML approach in both training time and performance. Our experiments also show promising results for highly imbalanced datasets. Moreover, all our experimentation was done on realistic benchmark datasets. In future we look to develop a new novel model architecture as well develop a novel aggregation scheme that would surpass the state of the art DL/ML approaches as well the existing FL based works on non-IID, imbalanced environments.

REFERENCES

[1] kingma2017adamKingma, D. & Ba, J. Adam: A Method for Stochastic Optimization. (2017)

[2] 4633969He, H., Bai, Y., Garcia, E. & Li, S. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. *2008 IEEE International Joint Conference On Neural Networks (IEEE World Congress On Computational Intelligence)*. pp. 1322-1328 (2008)

[3] 8466360Tang, F., Mao, B., Fadlullah, Z. & Kato, N. On a Novel Deep-Learning-Based Intelligent Partially Overlapping Channel Assignment in SDN-IoT. *IEEE Communications Magazine*. **56**, 80-86 (2018)

[4] pmlr-v54-mcmahan17aMcMahan, B., Moore, E., Ramage, D., Hampson, S. & Arcas, B. Communication-Efficient Learning of Deep Networks from Decentralized Data. *Proceedings Of The 20th International Conference On Artificial Intelligence And Statistics*. **54** pp. 1273-1282 (2017,4,20), https://proceedings.mlr.press/v54/mcmahan17a.html

[5] taheri2020fedTaheri, R., Shojafar, M., Alazab, M. & Tafazolli, R. FED-IIoT: A robust federated malware detection architecture in industrial IoT. *IEEE Transactions On Industrial Informatics*. **17**, 8442-8452 (2020)

[6] cheng2020federatedCheng, Y., Liu, Y., Chen, T. & Yang, Q. Federated learning for privacy-preserving AI. *Communications Of The ACM*. **63**, 33-36 (2020)

[7] mothukuri2021federatedMothukuri, V., Khare, P., Parizi, R., Pouriyeh, S., Dehghantanha, A. & Srivastava, G. Federated-learning-based anomaly detection for iot security attacks. *IEEE Internet Of Things Journal*. **9**, 2545-2554 (2021)

[8] 9064731Gong, M., Xie, Y., Pan, K., Feng, K. & Qin, A. A Survey on Differentially Private Machine Learning [Review Article]. *IEEE Computational Intelligence Magazine*. **15**, 49-64 (2020)

[9] AGRAWAL2022346Agrawal, S., Sarkar, S., Aouedi, O., Yenduri, G., Piamrat, K., Alazab, M., Bhattacharya, S., Maddikunta, P. & Gadekallu, T. Federated Learning for intrusion detection system: Concepts, challenges and future directions. *Computer Communications*. **195** pp. 346-361 (2022), https://www.sciencedirect.com/science/article/pii/S0140366422003516

[10] AGRAWAL2022346Agrawal, S., Sarkar, S., Aouedi, O., Yenduri, G., Piamrat, K., Alazab, M., Bhattacharya, S., Maddikunta, P. & Gadekallu, T. Federated Learning for intrusion detection system: Concepts, challenges and future directions. *Computer Communications*. **195** pp. 346-361 (2022), https://www.sciencedirect.com/science/article/pii/S0140366422003516

[11] briggs2020federatedBriggs, C., Fan, Z. & Andras, P. Federated learning with hierarchical clustering of local updates to improve training on non-IID data. *2020 International Joint Conference On Neural Networks (IJCNN)*. pp. 1-9 (2020)

[12] qin2020lineQin, Q., Poularakis, K., Leung, K. & Tassiulas, L. Line-speed and scalable intrusion detection at the network edge via federated learning. *2020 IFIP Networking Conference (Networking)*. pp. 352-360 (2020)

[13] fan2020iotdefenderFan, Y., Li, Y., Zhan, M., Cui, H. & Zhang, Y. Iotdefender: A federated transfer learning intrusion detection framework for 5g iot. *2020 IEEE 14th International Conference On Big Data Science And Engineering (BigDataSE)*. pp. 88-95 (2020)

[14] chen2020asynchronousChen, Y., Ning, Y., Slawski, M. & Rangwala, H. Asynchronous online federated learning for edge devices with non-iid data. *2020 IEEE International Conference On Big Data (Big Data)*. pp. 15-24 (2020)

[15] han2019federatedHan, Y., Li, D., Qi, H., Ren, J. & Wang, X. Federated learning-based computation offloading optimization in edge computing-supported internet of things. *Proceedings Of The ACM Turing Celebration Conference-China*. pp. 1-5 (2019)

[16] 9091574Latif, S., Zou, Z., Idrees, Z. & Ahmad, J. A Novel Attack Detection Scheme for the Industrial Internet of Things Using a Lightweight Random Neural Network. *IEEE Access*. **8** pp. 89337-89350 (2020)

[17] 9103025Fuller, A., Fan, Z., Day, C. & Barlow, C. Digital Twin: Enabling Technologies, Challenges and Open Research. *IEEE Access*. **8** pp. 108952-108971 (2020)

[18] 10.1145/3298981Yang, Q., Liu, Y., Chen, T. & Tong, Y. Federated Machine Learning: Concept and Applications. *ACM Trans. Intell. Syst. Technol.*. **10** (2019,1), https://doi.org/10.1145/3298981

[19] kairouz2021advancesKairouz, P., McMahan, H., Avent, B., Bellet, A., Bennis, M., Bhagoji, A., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R. & Others Advances and open problems in federated learning. *Foundations And Trends® In Machine Learning*. **14**, 1-210 (2021)

[20] lyu2022privacyLyu, L., Yu, H., Ma, X., Chen, C., Sun, L., Zhao, J., Yang, Q. & Philip, S. Privacy and robustness in federated learning: Attacks and defenses. *IEEE Transactions On Neural Networks And Learning Systems*. (2022)

[21] pmlr-v80-yin18aYin, D., Chen, Y., Kannan, R. & Bartlett, P. Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates. *Proceedings Of The 35th International Conference On Machine Learning*. **80** pp. 5650-5659 (2018,7,10), https://proceedings.mlr.press/v80/yin18a.html

[22] karimireddy2020byzantineKarimireddy, S., He, L. & Jaggi, M. Byzantine-robust learning on heterogeneous datasets via bucketing. *ArXiv Preprint ArXiv:2006.09365*. (2020)

[23] zhao2018federatedZhao, Y., Li, M., Lai, L., Suda, N., Civin, D. & Chandra, V. Federated learning with non-iid data. *ArXiv Preprint ArXiv:1806.00582*. (2018)

[24] li2020federatedLi, T., Sahu, A., Zaheer, M., Sanjabi, M., Talwalkar, A. & Smith, V. Federated optimization in heterogeneous networks. *Proceedings Of Machine Learning And Systems*. **2** pp. 429-450 (2020)

[25] sattler2020clusteredSattler, F., Müller, K. & Samek, W. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE Transactions On Neural Networks And Learning Systems*. **32**, 3710-3722 (2020)

[26] ghosh2019robustGhosh, A., Hong, J., Yin, D. & Ramchandran, K. Robust federated learning in a heterogeneous environment. *ArXiv Preprint ArXiv:1906.06629*. (2019)