Intrusion detection for Softwarized Networks with Semi-supervised Federated Learning

Ons Aouedi, Kandaraj Piamrat, Guillaume Muller, Kamal Singh

Intrusion detection for Softwarized Networks with Semi-supervised Federated Learning

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

***Abstract*—With the increasing development of 5G/Beyond 5G and network softwarization techniques, we have more flexibility and agility in the network. This can be exploited by Machine Learning (ML) to integrate intelligence in the network and improve network as well as service management in edge-cloud environment. Intrusion detection systems (IDS) is one of the challenging issues for managing network. However, traditional approaches in this domain require all data (and their associated labels) to be centralized at the same location. In this context, such approaches lead to: (i) a large bandwidth overhead, as raw data needs to be transmitted to the server, (ii) low incentives for devices to send their private data, and (iii) large computing and storage resources needed on the server side to label and treat all this data. In this paper, to cope with the above limitations, we propose a semi-supervised federated learning model for IDS. Moreover, we use network softwarisation for automation and deployment. Our model combines Federated Learning and Semi-Supervised Learning where the clients train unsupervised models (using unlabeled data) to learn the representative and low-dimensional features and the server conducts a supervised model (using labeled data). We evaluate this approach on the well-known UNSW-NB15 dataset and the experimental results demonstrate that our approach can achieve accuracy and detection rates up to 84.32% and 83.10%, respectively while keeping the data private with limited overhead.**

***Index Terms*—Federated Learning, Semi-supervised learning, Deep Learning, Machine Learning, Auto-Encoder, Intrusion Detection.**

I. Introduction

Recent development in smart devices has led to an explosion in data generation and heterogeneity. For example, by 2023, a 5G connection will generate nearly *3X* more traffic than a 4G connection [1]. These create significant challenges for the network operator, especially, for the security of the end-users. Consequently, to improve network security, some ultra-efficient, fast, and autonomous network management approaches are crucial. In this context, some advances such as network softwarisation and Machine Learning (ML)/Deep Learning (DL) can greatly improve the intrusion detection task and makes the network environment more flexible. Additionally, network sofwarization has brought flexibility and increased automation for network management and deployment. The Intrusion Detection System (IDS) functionality can be implemented as a virtual network function (VNF) and deployed at different levels on the infrastructure. The DL/ML models can be used to detect intrusions by analyzing the traffic behavior (normal/abnormal). However, using these models requires a central entity to process and label the data collected from all users in the network, which can introduce a privacy leakage of the sensitive data as well as network congestion. Also, IDS requires fast analysis whereas sending user data to some central server is time-consuming [2]. To cope with these issues, Federated Learning (FL) has been proposed as a decentralized machine learning scheme. It was firstly developed by Google [3]. FL means *joint learning* where several clients work together to train the models collaboratively [4]. One of the strong assumptions behind current FL-based solutions for IDS is that all the data is labeled [2]. However, since new types of attacks emerge every day, labeling data is expensive and often impossible [5]. Therefore, semi-supervised FL can be a promising solution to decrease the cost, of labeling all network traffic, by taking advantage of both labeled and unlabeled data [6].

In this paper, we propose Federated Learning with a semi-supervised approach for IDS, which combines semi-supervised learning with FL. Specifically, the FL clients and server train DL-based IDS models in a collaborative manner without exchanging data. In our scenario, we can imagine a cloud server running the FL server and edge nodes running FL clients. Therefore, by using our semi-supervised FL approach, the edge nodes will not need to label their local data. They will just learn the data representations through an unsupervised model (autoencoder). Then, unlike the classical FL, in our case, the FL server not only generates a global model but it exploits a small amount of labeled data to conduct supervised learning (Neural Network). The small amount of the labeled data can be provided by public/laboratory data without using the users' private data. As a result, our model can be improved with unlabeled data. FL will help to reduce the communication overhead, preserve the privacy of clients' local data, and avoid overloading the server with malicious/normal flows by sending the model parameters instead of users' raw data. Also, it avoids the time-wasting needed for labeling all the data.

In brief, our contributions can be summarized as follows:

- A semi-supervised FL, combining the use of unsupervised learning at the client and supervised learning at the server. The unsupervised and supervised models are then

concatenated to obtain a unified representation learning and classification solution for IDS automatically.
- Enabling the edge nodes to learn an efficient intrusion detection model without the need to label their local data.
- Decreasing the burden of transmitting and labeling all the traffic at the server: by using FL, we add the edge nodes into the pipeline of the learning process and employ unsupervised learning at the edge.

The rest of this paper is organized as follows. The related work is presented in Section II. Section III presents our solution with some backgrounds as well as its methodology and architecture. Section IV presents the datasets, experimental results, and performance of the proposed model under different configurations and compared with other well-known models. Finally, Section V concludes this paper.

II. Related work

Nowadays, IDS solutions are moving from using fixed sets of rules towards ML/data-based approaches. Such approaches allow to reduce the time of analysis, but also - and more importantly - to detect unknown attacks, as some ML solutions are self-learning. Various ML techniques have been used for IDS: Support Vector Machines (SVMs) [7]; ensemble learning (XGBoost [8, 9], Random Forests [10]), and DL [11, 12]. However, ML techniques in this context bring their own challenges and FL [3] (detailed in section III-A2) can solve some of them. With FL, data and labels do not need to be centralized at the same place hence there is no need to send huge amounts of raw data to a central server.

Chen *et al.* [13] propose an FL based IDS for IoT, based on Attention Gated Recurrent Units (GRU). They showed 8% better accuracy as compared to other centralized approaches, with a drastic drop of 70% in communication cost. This approach is particularly focused on and efficient against model poisoning attacks, but at the cost of introducing a bias in the selection of participating devices. Rahman *et al.* [14] propose an FL based IDS for IoT. They showed that FL achieves somewhat similar accuracy as centralized ML, but is better at protecting privacy and lowering network and computation overhead. It is also better than models learned directly on devices (using only local data), as FL benefits from others' data even without accessing them. Mothukuri *et al.* [15] propose an asynchronous FL system for IDS using an ensemble of GRU of various sizes, combined through a Random Forest model. The final model achieves accuracy similar to that of the centralized version, but with better privacy protection and a reduced number of false alarms. To detect attacks, they need to use recurrent NNs in order to extract relations between packets (end-to-end DL). Also, as in [14], they consider that the labels are directly accessible on the devices.

Though FL can be promising, the main drawback of the above approaches is the assumption that both data and labels are available. For labels, this assumption seems unrealistic, as this would mean that a human would have already tagged all the network traffic. This is difficult in practice (i) due to the resource constraints on the devices as well as some edge nodes and (ii) due to the difficulty of manually labeling data on such devices, which are far from human reach. To address this drawback, in this paper, we propose a novel semi-supervised approach, based on FL. The model consists of two parts, one trained on the edge nodes (unsupervised autoencoder, using unlabeled data) and the other on the cloud node (supervised classification, using labeled data).

III. Proposition

We present two main concepts used in our proposal (Autoencoder and FL), the methodology, the architectural design, and the potential implementation platform.

A. Background

1) Autoencoder (AE): AE is an unsupervised neural network, which can learn data representations. It consists of two parts: an encoder and a decoder. The encoder converts the input into an abstraction, which is generally known as a *code*, then the input can be reconstructed from the code layer through the decoder. It uses non-linear hidden layers to perform dimensionality reduction. For more details about AE, please refer to [16].

We use an AE because, unlike supervised deep neural networks, it is an unsupervised feature learning neural network that can extract features from unlabeled data automatically. Moreover, it is not a complex model and hence can be implemented on simple FL clients.

2) Federated Learning (FL): FL has been proposed in [3] as a solution to react with new networks' behavior (e.g., data heterogeneity and complexity). Within the FL concept, the data is maintained where it is generated and no raw data get exchanged. In other words, FL is a distributed machine learning concept where the data entities collaborate to jointly learn a global model without sacrificing the privacy of the clients. It is also more scalable than centralized ML and would be very useful in many aspects of network management today.

B. Methodology & Architecture

As mentioned earlier, getting a large amount of unlabeled traffic data is generally easy and does not require manual labeling; hence, our system takes advantage of both labeled and unlabeled data in a decentralized way and minimizes data exchange. It can be illustrated as in Figure 1.

- **FL Clients**

FL clients operate at the edge nodes since the end-users devices have limited capabilities (e.g., computing and storage resources). In other words, using the AE, the edge nodes conduct unsupervised learning and are able to capture the representative and low-dimensional features of the network traffic. Moreover, the edge nodes operate the IDS function using the *intrusion detection model* that was trained on the FL server to detect abnormal behavior.

- **FL server**

FL server operates on the cloud node. It does not only aggregate the different clients' AE but also trains the *supervised learning* model using labeled data. More concretely, after the

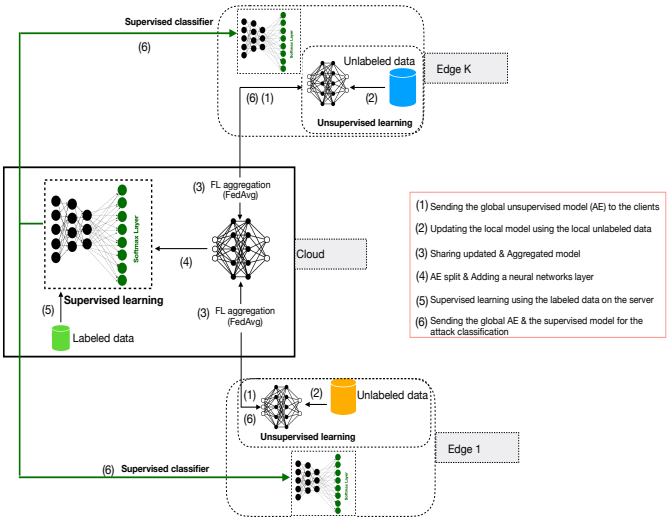

Fig. 1: Overview of the proposition

clients' AE aggregation, it removes the global decoder layers and directly links the global encoder layer to a neural network (NN) classifier. Then, the two models are concatenated together and the back-propagation algorithm is performed in order to fine-tune the deep supervised model for traffic classification.

The steps of the solution can be detailed as follows.

- **Step 1:** The FL server initiates the hyper-parameters of the AE model (*unsupervised learning*) and disseminates it to the edge nodes.
- **Step 2:** The selected edge nodes train the AE model using their *local unlabeled data*.
- **Step 3:** The FL server aggregates the weights from the different edge nodes' AE once all the updates are received. For the aggregation, the FederatedAveraging (FedAvg) algorithm [3] is used.

$$W_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} W_{t+1}^k \tag{1}$$

  where W is the model parameters at the iteration $t+1$, n is the total number of the unlabeled data in all the K clients, n_k is the number of training unlabeled data in client k.
- **Step 4:** The FL server deletes the global decoder and concatenates the global encoder part of the global autoencoder and the Neural Networks classifier.
- **Step 5:** The FL server uses the labeled data to fine-tune the whole concatenated model and hence gets a supervised model.
- **Step 6:** The FL server sends back the aggregated AE (*unsupervised model*), obtained in Step 3, to the edge nodes for new local training as well as the concatenated model for the IDS task.

In each communication round, steps **2, 3, 4, 5, and 6** are repeated for continuous learning and improvement.

C. Orchestration of our proposition over Cloud and Edge

In order to orchestrate our training workers and aggregation worker, a Kubernetes-based framework can be used: Akraino [1]. The training workers will run as containers on the edge nodes and will communicate with the aggregation worker running on the cloud node. Thus, we will be able to start the training and aggregation workers, configure the aggregation algorithm, configure the hyper-parameters, and scale according to the need. This will allow the automation and orchestration of our process. The IDS task will also run as a worker. It will detect attacks using the trained ML model in real-time. Once an improved ML model has been trained and validated then it will replace the previous model.

IV. Experiment and Results

In this section, we first present the dataset used in our experiments. Then, we present the model architectures used for intrusion detection. Finally, the results are analyzed and discussed.

A. Dataset description

In order to train our semi-supervised FL model, we use the UNSW-NB15 dataset [2] [17] because it is recent and referenced in many existing papers. The simulation period of data was 16 hours on Jan 22, 2015, and 15 hours on Feb 17, 2015. It has 9 types of attacks: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms. The training set contains 175,341 and the testing set contains 82,332 total observations. The simulation of a partially labeled dataset has been done through the random selection of a portion of labeled observations whose labels are removed. Before training the models, the data are normalized so all their values are in the range of [0, 1] in order to optimize the training process' performance.

TABLE I: Implementation parameters

Dataset	
Nb input variables	197
Nb output variables	1
Training set	175,341
Unlabeled set	122,739
Labeled set	52,602
Unlabeled ratio R_u	2.33
Testing set	82,332 (default)
Federated Learning	
FL server	1
Nb clients	100
Clients used in federated updates	10%
AE epochs (client)	5
NN epochs (server)	20
Communication round	6

[1]https://wiki.akraino.org/display/AK/Federated+Learning
[2]https://research.unsw.edu.au/projects/unsw-nb15-data-set

B. Results

In this section, we study the performance of the proposed semi-supervised FL for IDS under different factors such as model architecture, communication rounds, the ratio of unlabeled data, and communication overhead. Also, we compare our model with its non-FL version (using the same deep learning algorithms, amount of labeled/unlabeled data). Table I summarizes the parameters and their selected settings. In our experiments, to avoid the problem of client failure and communication overhead, we selected randomly a given number of clients (10%) to participate in each communication round. All experiments were run using four core Intel® Core™ i7-6700 CPU@3.40GHz processor, and 32.00 GB of RAM. The code is made available online [3].

1) Impact of model architectures: In DL, the selection of hyperparameters, such as the number of hidden layers and hidden units in each hidden layer, greatly affects the performance of the model and hence impacts the performance of the FL solution. Since there is no clear mathematical proof to interpret its architecture, we conduct several experiments in order to find the optimal model. Table II presents different model configurations and their performance in terms of accuracy. To find the optimal model, we started with simple architecture, and then we have progressively increased its complexity. From this table, we can notice that it is not true that when we increase the complexity of the model (i.e., hidden layers and number of neurons), we will get better results. As a consequence of the above analysis, the configuration in bold text with 3 hidden layers for the encoder parts **[150, 100, 50]** is selected for further experiments since it provides the best accuracy. It is important to note here, that since we used a symmetric AE, the decoder part has the same encoder architectures. Consequently, our supervised model consists of an input layer, 3 encoder layers, and NN layer.

TABLE II: Hidden layers configurations used in the experimentation

Model	#Hidden layers	Number of neurons				FL Test accuracy
		L1	L2	L3	L4	
M_1	3	50	50	30	-	80.58%
M_2	3	50	50	50	-	80.48%
M_3	3	100	100	50	-	78.05%
M_4	**3**	**150**	**100**	**50**	**-**	**84.32%**
M_5	4	150	100	80	50	80.46%
M_6	4	100	100	80	50	79.46%
M_7	4	150	110	70	50	78.08%

2) Impact of communication rounds: In this subsection, we study the relationship between the performance of the semi-supervised FL and the communication rounds. Since the edge nodes generally have limited resources compared to the cloud node, we set the number of epochs to 5 for the AE model on the edges and to 20 epochs for the supervised models located on the cloud. For each communication round (between FL server and clients), we present the performance of the proposed model (Figure 2) while keeping the remaining parameters fixed.

It can be seen from the results presented in Figures 2, that the accuracy goes from **82.82%** in the second round to **84.32%** in round 6. However, it can be seen also that increasing the number of communication rounds can decrease the performance of the model. This means that in our case, the final model can overfit when the number of communication rounds is higher than 6.

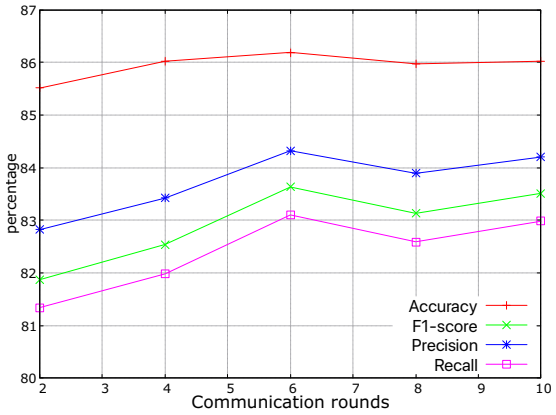

Fig. 2: Effect of communication rounds on the FL model performance

3) Trade-off between performance and unlabeled data: To investigate the impact of unlabeled data, we train our system using different ratios of unlabeled samples R_u while keeping the amount of labeled data fixed.

$$R_u = \frac{nb\ unlabeled\ data}{nb\ labeled\ data} \tag{2}$$

The performance on the testing set is presented in Figure 3. It can be seen that, as the amount of unlabeled samples increases, accuracy increases, and loss decreases rapidly. In particular, the accuracy of our model exceeds 80% when the R_u is more than 1.3. This can be explained by the fact that increasing training set size through adding a huge volume of unlabeled data can provide valuable information for the model. Specifically, the local data on the edge nodes can improve the performance of the AE and reduce the reconstruction error (finding more relevant representation), which in turn helps the supervised model to distinguish better between normal and malicious traffic.

4) Communication overhead: By only exchanging the local model updates between the FL server and the clients, FL can help to reduce the communication overhead. Therefore, to minimize the communication overhead, two key aspects need to be considered: (i) reducing the local's model update frequency, (ii) reducing the size of data communicated between the FL server and the clients [18]. We have taken into consideration these two aspects by considering two scenarios. The first scenario is when we use centralized learning and the second one is with FL. For the FL, we also consider

[3]https://github.com/aouedions11/Semi-supervised-Federated-Learning-for-IDS.git

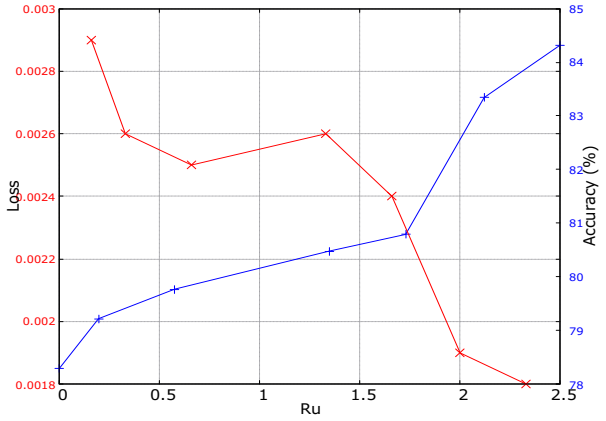

Fig. 3: Performance of model with different unlabeled ratio.

two other scenarios by changing AE frequency updates, in the first one (our model with an update every 5 epochs), the edge nodes update their local model (e.g., autoencoder) after 5 epochs, while in the second one (our model with an update every 20 epochs), the local models are updated after 20 epochs. We can observe from Figure 4 that the local model's update frequency can impact the communication overhead. Moreover, in comparison to the centralized scenario, our solution significantly reduces the messages' size. This advantage will become even more significant in the case of larger training data. This is mainly due to the fact that FL avoids transferring raw data samples to the central entity and sends only model parameters. Also, through the use of the autoencoder model, the FL clients compress their local data and hence reduce the size of the parameters communicated with the FL server.

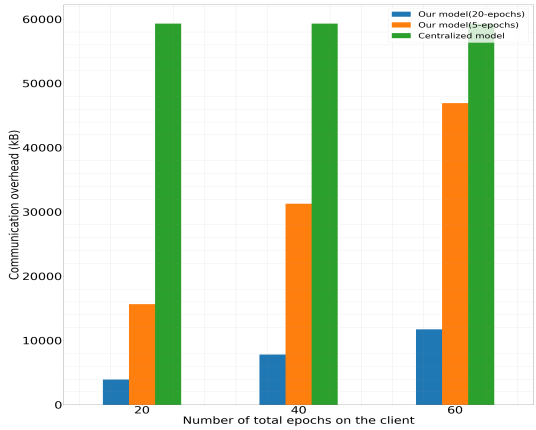

Fig. 4: Comparison in terms of communication overhead

5) Comparison with supervised models: To verify the classification efficiency of the proposed semi-supervised FL, we compared it to four reference ML models including (i) simple classifiers, which are Decision Tree (DT) and Support Vector Machine (SVM), (ii) ensemble learning such as Random Forest (RF). In fact, these models use only the labeled data located on the FL server for their learning process. As presented in Table I, the number of labeled samples is 52,602. It is important to note here, that we test the performance of these classifiers on the same test set used with our model. Table III and Figure 5 illustrate the comparison of our proposed model with those classifiers. It is worth noting that the semi-supervised federated learning model outperforms the classical supervised models in terms of all the evaluation metrics. For example, the F1-score is increased by **3.68%**, **5.46%**, **6.21%**, **7.55%** for MLP, RF, SVM, and DT, respectively. This may be attributed to the fact that the use of unlabeled data in the training process boosts the performance of our model. In addition, the proposed model outperforms these classifiers because, with the help of the clients' private data, the AE models generate deeply learned features that yield superior results compared to the initial statistical features.

TABLE III: Comparison with supervised models.

Model	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1-score (%)**
DT	76.48	76.36	75.94	76.08
SVM	79.46	85.94	77.21	77.42
RF	80.13	86.70	77.90	78.17
MLP	81.11	84.16	79.49	79.95
proposed model	**84.32**	**86.19**	**83.10**	**83.63**

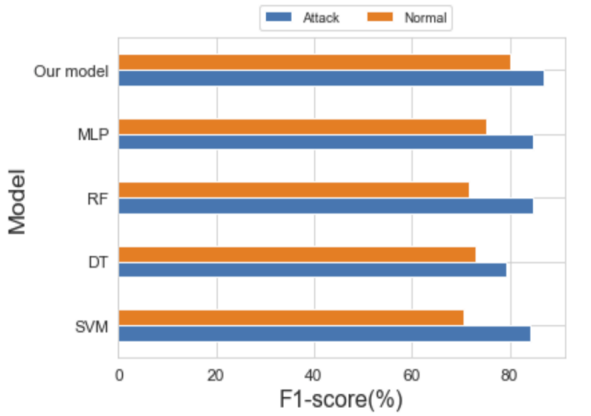

Fig. 5: The performance of identifying normal and attack flows of our model against supervised models.

6) Comparison with Non-FL model: In this section, we compare our model with the non-FL semi-supervised model in order to evaluate the performance of our model. Here the non-FL-based model is trained on the centralized training data using the same amount of unlabeled/labeled data, and DL architectures (e.g., layers, neurons) used during the training of our model. As shown in Table IV, our proposed model outperforms its non-FL version in terms of accuracy, precision, recall, and F1-score. This is maybe attributed to the training rounds between the FL server and the clients that improve the intrusion detection performance. These results illustrate the effectiveness of our proposed model without leaking users' private data.

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

	Accuracy	**Precision**	**Recall**	**F1-score**
Non-FL model	81.40	83.83	79.92	80.39
proposed model	**84.32**	**86.19**	**83.10**	**83.63**

V. Discussion and conclusion

In this paper, a semi-supervised, federated-learning, based Intrusion Detection System has been presented. This model was trained in a semi-supervised way, in which both the labeled and unlabeled data were used. Moreover, it overcomes the difficulties of central data storage, processing, and privacy concerns for sharing sensitive data, by keeping the data close to where it was generated. The experimental results demonstrate that our model gave good results thanks to the deep learning structure, collaborative learning combined with a feature extraction ability.

The presented results demonstrate that using a small amount of labeled data on the FL server helps to reduce the human-labor operation to annotate the clients' data. On the other hand, the support of unlabeled data for the training process on the edge enhances the performance of the learned model. Using our semi-supervised FL model, the edge networks have the ability to detect attacks without the need to have labeled samples or share their private data. Also in case of connection loss, the edge servers can still independently detect anomalies in their edge network since the models are locally located. In addition, the FL server benefits from the FL clients to improve its supervised model without the need to collect or process 100% data in a centralized way. Finally, using FL helps to reduce the communication overhead and hence reduce the network congestion if all traffic has to be sent to the FL server.

For future works, we plan to implement the model in Akraino and study the network performance as well as improve the accuracy of the proposed model and cross verifying with other datasets.

References

[1] "Cisco visual networking index: Global mobile data traffic forecast update, 2017–2022 white paper, cisco, 2019.."

[2] S. Agrawal, S. Sarkar, O. Aouedi, G. Yenduri, K. Piamrat, S. Bhattacharya, P. K. R. Maddikunta, and T. R. Gadekallu, "Federated learning for intrusion detection system: Concepts, challenges and future directions," *arXiv preprint arXiv:2106.09527*, 2021.

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

[4] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, and Y. Zhou, "A hybrid approach to privacy-preserving federated learning," in *ACM Workshop on Artificial Intelligence and Security*, pp. 1–11, 2019.

[5] Z. Zhang, Y. Yang, Z. Yao, Y. Yan, J. E. Gonzalez, and M. W. Mahoney, "Improving semi-supervised federated learning by reducing the gradient diversity of models," *arXiv preprint arXiv:2008.11364*, 2020.

[6] Y. Zhao, H. Liu, H. Li, P. Barnaghi, and H. Haddadi, "Semi-supervised federated learning for activity recognition," *arXiv preprint arXiv:2011.00851*, 2020.

[7] F. E. Heba, A. Darwish, A. E. Hassanien, and A. Abraham, "Principle components analysis and support vector machine based intrusion detection system," in *2010 10th international conference on intelligent systems design and applications*, pp. 363–367, IEEE, 2010.

[8] P. Kumar, G. P. Gupta, and R. Tripathi, "An ensemble learning and fog-cloud architecture-driven cyber-attack detection framework for iomt networks," *Computer Communications*, 2020.

[9] A. Husain, A. Salem, C. Jim, and G. Dimitoglou, "Development of an efficient network intrusion detection model using extreme gradient boosting (xgboost) on the unsw-nb15 dataset," in *2019 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, pp. 1–7, IEEE, 2019.

[10] J.-H. Lee and K. Singh, "Switchtree: in-network computing and traffic analyses with random forests," *Neural Computing and Applications*, pp. 1–12, 2020.

[11] G. Pang, C. Shen, L. Cao, and A. v. d. Hengel, "Deep learning for anomaly detection: A review," *arXiv preprint arXiv:2007.02500*, 2020.

[12] Q. Niyaz, W. Sun, and A. Y. Javaid, "A deep learning based ddos detection system in software-defined networking (sdn)," *arXiv preprint arXiv:1611.07400*, 2016.

[13] 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*, vol. 8, pp. 217463–217472, 2020.

[14] S. A. Rahman, H. Tout, C. Talhi, and A. Mourad, "Internet of things intrusion detection: Centralized, on-device, or federated learning?," *IEEE Network*, vol. 34, no. 6, pp. 310–317, 2020.

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

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

[17] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 military communications and information systems conference (MilCIS)*, pp. 1–6, IEEE, 2015.

[18] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.