Privacy-Preserving Defense: Intrusion Detection in IoT using Federated Learning

Leonardo Almeida*§, Pedro Rodrigues*§, Rafael Teixeira*, Mário Antunes*† and Rui L. Aguiar*†
*Instituto de Telecomunicações, Universidade de Aveiro, Aveiro, Portugal
E-mails: {leonardoalmeida,pedrofrodrigues,rafaelgteixeira,mario.antunes,ruilaa}@av.it.pt
†DETI, Universidade de Aveiro, Aveiro, Portugal
E-mails: {mario.antunes,ruilaa}@ua.pt

Abstract—**The Internet of Things (IoT) presents unprecedented challenges in network security due to its vast deployment and resource limitations. Addressing these challenges requires robust Intrusion Detection Systems (IDS) capable of detecting and mitigating potential threats effectively. In this study, we explore the efficacy of Federated Learning (FL) in training IDS models while ensuring data privacy in IoT scenarios. We leverage three state-of-the-art datasets to evaluate FL-based training approaches with varying numbers of workers. Our experiments demonstrate that FL-based training yields comparable performance to traditional single training methods across multiple performance metrics. Additionally, FL training exhibits faster convergence times, highlighting its efficiency and scalability for training IDS models in IoT environments. These findings underscore the potential of FL as a privacy-preserving technique for enhancing network security in IoT deployments.**

Index Terms—**IoT Security, Intrusion Detection Systems, Machine Learning, Neural Networks, Federated Learning, Decentralized Security**

I. Introduction

The Internet of Things (IoT) is a network of physical objects connected to the Internet and able to communicate with each other. The IoT devices are used in various applications such as smart homes, smart cities, healthcare, transportation, etc. The affordability and availability of these devices made them a popular candidate for the development of solutions for real-world problems, leading to an exponential growth of IoT networks.

The lack of security in IoT devices due to their limited resources and the constant increase in the complexity of IoT networks makes them more difficult to monitor and protect, creating a new challenge for the security community.

An effective way to protect IoT networks is to use Intrusion Detection Systems (IDS) based on Machine Learning (ML) [1]. The use of ML algorithms in IDS is a promising approach to detect and prevent attacks in IoT networks by classifying the network traffic as normal or malicious, and in case of malicious traffic, identifying the type of attack. To develop ML models capable of detecting attacks in this type of network, it is necessary to have a large dataset that contains network traffic data from IoT devices. However, most of the data generated within these networks is private and not available for research purposes, as it is mostly generated within personal or enterprise environments.

†These authors contributed equally to this work.

One way to solve this problem is by generating synthetic data that resembles the real data. Nevertheless, this approach is not ideal, as the synthetic data may not be representative of real-world scenarios, and the models trained with this data may not be able to generalize to the real data [2]. Another alternative, and the one explored in this article, is the adoption of distributed techniques to train ML models with data from multiple sources without the need to share the private data with a central server.

Federated Learning (FL) is one of the distributed learning techniques that stands as a crucial paradigm in ensuring data privacy in the contemporary landscape of ML. By distributing model training across decentralized devices or servers, FL mitigates the risks associated with centralizing sensitive data in one location [3]. This approach allows individual user data to remain on their respective devices, thereby reducing the likelihood of breaches or unauthorized access to personal information.

This article demonstrates the relevance of training ML models for Intrusion Detection Systems using FL techniques, guaranteeing the privacy of the data generated within the IoT networks. To achieve this goal, we will study the performance of different ML models, first using a completely centralized approach and then using a decentralized approach based on FL. In order to increase the feasibility of this study, we will use three IDS state-of-the-art datasets, which contain network traffic data from IoT devices. The results obtained will allow us to compare the performance of the different ML models and to evaluate the impact of the FL technique on the performance of the models.

The remaining sections of this document are organized as follows: Section 2 presents the related work. Section 3 describes the datasets used in this study. Sections 4, 5 and 6 describe and present the ML models, the FL approach followed, and the experiments carried out, respectively. Section 7 presents the results and discussion. Finally, Section 8 presents the conclusions and future work.

II. Related Work

There is a vast body of research available exploring diverse IDS in various datasets for IoT networks. Notably, these studies often demonstrate impressive model performance. While individual works employ differing data and models, their findings contribute significantly to our research due to their shared focus on relevant themes and the use of similar datasets. Namely, the use of Deep Neural Network (DNN) is a common theme in the literature, as it is a powerful tool for capturing complex patterns within network data.

For example, in [4], the authors employed a Feed-Forward Deep Neural Network (FFDNN) enhanced with a Wrapper Based Feature Extraction Unit (WFEU) to analyze two datasets: UNSW-NB15 and AWID. This approach achieved an overall accuracy of 87.10% for binary classification and 77.16% for multiclass classification on the UNSW-NB15 dataset. On the AWID dataset, the model demonstrated even higher accuracy, reaching 99.66% for binary classification and 99.77% for multiclass.

Similarly, a DNN is used in [5]. The model combines a Convolutional Neural Network (CNN) for feature extraction from large datasets with a Weight-dropped Long Short-Term Memory (WDLSTM) network. The WDLSTM captures long-term dependencies within those features, mitigating overfitting risks inherent in recurrent connections. Evaluated on the UNSW-NB15 dataset, the model achieved impressive accuracy, reaching 98.43% for binary classification and 97.17% for multiclass classification.

In [6], the authors presented a kNN classifier empowered by an enhanced Arithmetic Optimization Algorithm (AOA) in their work. Seeking to refine accuracy, they implemented a parallel approach facilitating increased communication among populations and utilized a Lévy flight strategy for optimized updates. This "PL-AOA" algorithm, evaluated on the WSN-DS dataset, achieved an impressive 99% accuracy.

The model used in [7] is a bidirectional Long-Short-Term-Memory (BiDLSTM) deep learning model for intrusion detection. Trained and tested on the NSL-KDD dataset, the model demonstrated exceptional training accuracy (99.95%), although testing accuracy dipped to 94.26%. Notably, it showcased superior detection accuracy for U2R and R2L attacks compared to conventional LSTMs and boastingly low false alarm rates outperforming existing models.

In [8], the authors propose a hybrid model that combines a CNN and a Bidirectional Long Short-Term Memory (BiLSTM) for network intrusion detection. The model was trained and tested on the NSL-KDD and UNSW-NB15 datasets and achieved an accuracy of 83.58% and 77.16%, respectively. To overcome the problem of data imbalance, the authors used the one-side selection (OSS) to reduce the noise samples in the majority category and then increase the minority samples by Synthetic Minority Over-sampling Technique (SMOTE).

In [9], the authors explored XGBoost, a powerful algorithm for handling imbalanced multiclass classification, for IoT IDS. Tested on two datasets (X-IIoTDS and TON_IoT), the model achieved exceptional F1 scores of 99.9% and 99.87%, respectively. XGBoost was used because of its advantages, including learning from its mistakes, fine-tuning extensive hyperparameters, scaling imbalanced data, and processing null values.

The model proposed in [10] is a "Hybrid Convolutional Recurrent Neural Network-Based Network Intrusion Detection System (HCRNNIDS)". This system leverages the strengths of both CNNs and Recurrent Neural Networks (RNNs). By employing CNNs for extracting local features and RNNs for capturing temporal dependencies, HCRNNIDS significantly enhances the performance and prediction capabilities of traditional intrusion detection systems. Evaluated on the CSE-CIC-DS2018 dataset using 10-fold cross-validation, the model achieved an impressive accuracy of up to 97.75%.

The authors in [11] used a CNN for anomaly-based intrusion detection in IoT networks. This deep learning approach effectively analyzed traffic data from two datasets (NID and BoT-IoT), achieving impressive accuracy rates of 99.51% and 92.85%, respectively. The CNN model acted as a powerful tool for proactively identifying potential intrusions and anomalous behaviors within IoT network traffic.

In [12], the authors developed an efficient hybrid network-based IDS model (HNIDS), which is combined with an enhanced genetic algorithm and particle swarm optimization (EGA-PSO) and improved random forest (IRF) methods. This model is capable of dealing with the data-imbalance issues and was tested on the NSL-KDD dataset, achieving an accuracy of 88.149% and 98.979% for the binary and multiclass classification, respectively.

In [13], the authors proposed an effective approach for intrusion detection by combining a Support Vector Machine (SVM) with a novel naïve Bayes feature embedding technique. This method was tested on four diverse datasets: UNSW-NB15, CICIDS2017, NSL-KDD, and Kyoto 2006+. Notably, the model achieved impressive accuracy rates across all datasets, ranging from 93.75% to 99.35%. This success can be attributed to the data quality improvement achieved through the naïve Bayes embedding, ultimately leading to a more robust and less complex intrusion detection model.

Table I summarises the IDS state-of-the-art, highlighting the dataset and ML model used, where it is clear that current IDS approaches lean heavily towards the use of DNNs. These approaches showcase the adaptability and effectiveness of DNN in capturing complex patterns within network data, making them a leading choice in the field of IDS.

The current state of IDS leans heavily towards the utilization of DNN. Various models have demonstrated impressive performances across diverse datasets.

III. Datasets

There are several well-known datasets that could be used to train and test IDS models, for example, the KDD98 [14], KDDCUP99 [15], or NSLKDD [16]. However, as highlighted in [17], even though these and other similar datasets are sufficiently large and trusted by the research community, they are outdated and do not inclusively reflect today's network traffic, which seriously limits their applicability within contemporary studies of network security.

TABLE I: Summary of the IDS state-of-the-art.

REF	Datasets	Models
[4]	UNSW-NB15 & AWID	DNN
[5]	UNSW-NB15	DNN
[6]	WSN-DS	kNN
[7]	NSL-KDD	DNN
[8]	NSL-KDD & UNSW-NB15	DNN
[9]	X-IIoTDS & TON_IoT	XGBoost
[10]	CSE-CIC-DS2018	DNN
[11]	NID & BoT-IoT	DNN
[12]	NSL-KDD	Random Forest
[13]	UNSW-NB15, NSL-KDD, CICIDS2017 & Kyoto 2006+	SVM

Taking this into account, we decided to use three recent and widely used state-of-the-art datasets, namely UNSW-NB15 [18], ToN-IoT [19], and IoT-DNL[1], which contain network traffic data from IoT devices.

A. UNSW-NB15

The UNSW-NB15 dataset is a synthetically generated dataset created by the Australian Centre for Cyber Security (ACCS). This dataset has a hybrid of the real modern normal and the contemporary synthesized attack activities of the network traffic. Comprising over 2 million records with 49 features, the UNSW-NB15 dataset is prepared to train both binary and multi-class classification models since each malicious activity is additionally labeled with one of the 9 attack categories, namely DoS, Analysis, Backdoor, Exploits, Fuzzers, Generic, Reconnaissance, Shellcode, and Worms.

To avoid complex preprocessing procedures, in this study, we used the NF-UNSW-NB15[2] preprocessed dataset published in [20] by the University of Queensland. This version of the dataset incorporates the exact same information as the original dataset but uses NetFlow features. NetFlow is a popular network protocol developed by Cisco for collecting IP traffic information and monitoring network flow. According to the authors of this version of the dataset, the motivation behind this version is the lack of a standard feature set in general IDS datasets.

B. ToN-IoT

The ToN-IoT is a heterogeneous dataset also created by the ACCS and released in 2019. It is a comprehensive dataset that includes the telemetry data of an IoT network. Several portions of the dataset contain different traces of IoT services, network traffic, and operating system logs. The data was generated using a realistic network testbed. The dataset contains several attack scenarios such as Backdoor, Denial-of-Service (DoS), Distributed DoS (DDoS), Injection, Man In The Middle (MITM), Password, Ransomware, Scanning, and Cross-Site Scripting (XSS). Similarly to the UNSW-NB15 dataset, the ToN-IoT dataset can be used not only for multi-class classification but also for binary classification able to distinguish between normal and malicious traffic.

For this study, we used the NetFlow version of the ToN-IoT dataset, NF-ToN-IoT[3], published by the same authors as the NF-UNSW-NB15 dataset.

C. IoT-DNL

IoT-Device-Network-Logs (IoT-DNL) is a dataset tailored for network-based Intrusion Detection Systems. This open-source dataset, initially introduced by Sahil Dixit and sourced from Kaggle, is specifically designed for evaluating anomaly-based IDS in wireless environments. The dataset is flow-based and labeled, focusing on IoT devices, and has undergone preprocessing to suit the requirements of network-based IDS.

The network logs in this dataset were gathered by monitoring the network using ultrasonic sensors, more specifically, the Arduino and NodeMCU with the ESP8266 Wi-Fi module. Then these devices transmit data to the server via Wi-Fi. In total, the dataset comprises 477,426 samples, each with 14 features. A sample of the dataset could be classified as normal or malicious, and if malicious, it could be further classified into one of the following five categories: Wrong setup, DDoS, Data Type Probing, Scan, and MITM.

IV. Models

Given the popularity of Neural Networks (NNs) for IDS, we decided to use NNs established and tested in the literature for these datasets, given their proven performance and the fact that NNs are well-suited for FL.

Starting with the UNSW-NB15 dataset, in [21], the authors use 3 different NNs, including Artificial Neural Network (ANN), CNN, and Recurrent Neural Network (RNN). All models performed well and achieved similar results, so we decided to use the ANN model, which is the simplest and fastest model to train. The model consists of 3 hidden layers with 128, 96, and 64 neurons, respectively, and a dropout layer with a dropout rate of 0.25 after the hidden layers. The activation function used in the hidden layers is the Rectified Linear Unit (ReLU) function, and the optimizer used is the Adam optimizer.

The ToN-IoT dataset was used in [22] with a Sparse AutoEncoder (SAE) and a ANN model. The ANN consists of 2 hidden layers with 32 and 16 neurons, respectively, and a dropout layer after each hidden layer with a dropout rate of 0.2. The activation function used in the hidden layers is the ReLU function. To train the model, the authors used the Adam optimizer.

In [23], the authors explore the IoT-DNL dataset with a ANN model but do not specify the number of neurons in each layer. To replicate their model, we evaluated 18 different configurations of hidden layers and neurons and selected the

[1]https://www.kaggle.com/datasets/speedwall10/iot-device-network-logs
[2]https://espace.library.uq.edu.au/view/UQ:ffbb0c1
[3]https://espace.library.uq.edu.au/view/UQ:38a2d07

best-performing one. The model consists of 4 hidden layers with 64 neurons each and a dropout layer with a dropout rate of 0.1 between each hidden layer. The activation function used in the hidden layers is the ReLU function, and the model was trained using the Adam optimizer. Our evaluation of the model was made taking into account training and validation accuracy, and the complexity of the model.

For each model, the input and output layers were adapted to the number of features and classes of each dataset.

V. Federated-Learning

Before we present the details of the FL approach considered in this study, it is essential to understand the fundamental concepts of the broader field of Distributed Machine Learning. Distributed Machine Learning aims at *scaling out* to train large models using the combined resources of clusters of independent machines. Depending on the specific use case, the training process can follow different principles, techniques, and architectures. A well-known taxonomy proposed in [24] classifies distributed Machine Learning based on the following dimensions:

- **Model vs. Data Parallelism**: In model parallelism, the model is divided into partitions, which are then processed on distinct machines. To perform inference or train the model, signals have to be transmitted so that each partition can be evaluated. On the other hand, the main idea behind data parallelism is to distribute the data across the machines and then replicate the model onto each machine, where backpropagation can be performed in parallel to gather information about the loss function faster. In a general sense, first, each cluster node downloads the current model, then each node performs backpropagation using its assignment of data in parallel, and finally, the respective results are aggregated and integrated to form a new model.
- **Centralized vs. Decentralized Optimization**: In centralized optimization, a single optimizer instance, often referred to as a *parameter server*, handles the updating of specific model parameters, while other nodes focus on obtaining gradients through backpropagation. Because the parameter server is the only one with write access to a particular model parameter, its state consistently reflects the ongoing training progress. Furthermore, for workers to generate relevant gradients, they must frequently re-download the model. In contrast, decentralized optimization allows each node to update the model parameters independently, and the model is updated by aggregating the local updates. Following this technique, each worker acts as an independent learner who repeatedly observes the loss function and adjusts its local model parameters to decrease its loss.
- **Synchronous vs. Asynchronous Scheduling**: In synchronous scheduling, all workers have to wait for the slowest worker to finish their task before the next iteration can start. This approach is particularly useful when the workers have similar computational power and their communication is reliable. On the other hand, asynchronous scheduling allows each worker to proceed with the next iteration as soon as it finishes its current tasks. In other words, the workers are allowed to operate at their own pace. This approach is particularly useful when the workers have different computational power or the communication with the parameter server is unreliable.

FL is a specific instance of Distributed Machine Learning where the model is trained across multiple decentralized consumer-grade devices or servers holding local data samples without exchanging them. It enables privacy-preserving training as it does not require data to leave the local devices, which makes this technique a promising approach to training IDS models. However, as concluded in [25], similar to any machine learning approach, FL will not always be the best approach to a problem as its results highly depend on the devices and connection them.

For this study, the FL approach considered follows the principles of data parallelism, decentralized optimization, and asynchronous scheduling. Here the training process is orchestrated by a parameter server, which is responsible for aggregating the local model updates and broadcasting the global model to the workers. The workers, in turn, are responsible for training the model using their local data samples and sending the local model updates to the server. The server then aggregates the local model updates and broadcasts the global model to the workers. This process is repeated until the defined number of global epochs is reached.

VI. Experimental methodology

To evaluate the performance of the models and the impact of the FL technique, we conducted a series of experiments using the three datasets. The code used to obtain the results presented is available on Github[4].

The first step was to improve the preprocessed data. To do so, we removed irrelevant features, then normalized the data and split it into training, validation, and test sets. Based on [26], to avoid bias towards attacking/victim nodes and applications, for the UNSW-NB15 and ToN-IoT datasets, we removed the features “IPV4_SRC_ADDR”, “L4_SRC_PORT”, “IPV4_DST_ADDR”, “L4_DST_PORT” and “Attack”, making a total of 39 features. Furthermore, we normalized the data using the Quantile Transformer. Given the large size of these datasets, we only used 25% of the data, keeping the same proportion of normal and malicious traffic. For the IoT-DNL dataset, we removed the features “frame.number” and “frame.time”, making a total of 11 features and then normalized the data using the Standard Scaler. All datasets were split into training, validation, and test sets using a 64/16/20 ratio, respectively. In Table II, the number of samples of each dataset is presented.

For each dataset, four different experiments were conducted. The first experiment consisted of training the model without the FL technique (single training). The other three experiments

[4]https://github.com/leoalmPT/FL-analysis

TABLE II: Preprocessed datasets.

Dataset	Total Samples	Training Samples	Validation Samples	Test Samples
UNSW-NB15	2390275	382444	95611	119514
ToN-IoT	16940496	2710479	677620	847025
IoT-DNL	477426	305552	76388	95486

trained the model using the FL technique with 2, 4, and 8 workers, respectively. The training data was distributed equally among the workers. The models were trained on the training set and validated on the validation set. The test set was used to evaluate the performance of the models on unseen data. During the training process, three metrics were monitored: F1 Score, Accuracy, and Matthew's Correlation Coefficient (MCC). To avoid overfitting and to speed up the training process, an early stopping mechanism was implemented, which consists of stopping the training process when the validation MCC did not improve for 5 consecutive epochs. After the training process, the weights of the epoch where the model achieved the best validation MCC were restored. Regarding the hyperparameters, the models were trained using the Adam optimizer with a learning rate of 0.0001 and batch size of 1024. In the case of single training, the models were trained for a maximum of 100 epochs, while in the case of FL training, the models were trained for a maximum of 100 global epochs with 3 local epochs. Because of the early stopping mechanism, the number of global epochs was, in most cases, between 15 and 30.

VII. Results and discussion

The main results for the models' performance are presented in Table III. The results show that performance for the training, validation, and test data are consistent across all experiments, which means that the models did not overfit the training data and were able to generalize to the test data. Overall, the models trained using the FL technique with different numbers of workers achieved similar performance to the models trained in a single training. The only model that underperformed was FL with 2 workers in the ToN-IoT dataset, achieving a 0.8270 MCC, compared to 0.9252 with 8 workers. This result is likely due to the workers converging to different local minima or due to the early stopping mechanism that stopped the training earlier due to a lack of improvement in the last five epochs, while the higher number of workers helps mitigate this fact. Other metrics were evaluated (F1-Score and Accuracy), with results comparable to the ones obtained with MCC; however, due to the length restrictions of the paper, the results will not be presented here. Nonetheless, the results are available in the GitHub repository.

Another important aspect to consider is the training time. Table IV shows the time for training and validating the models without any communication restriction. The experiments were conducted on a single machine with an Intel Core i7-13700KF CPU and 64GB of RAM, where different workers are assigned

TABLE III: Experiments MCC

Dataset	Model	Training MCC	Validation MCC	Test MCC
UNSW-NB15	Single training	0.9533	0.9531	**0.9492**
	2 workers	0.9530	0.9513	**0.9503**
	4 workers	0.9520	0.9507	0.9487
	8 workers	0.9499	0.9476	0.9471
ToN-IoT	Single training	0.9453	0.9450	**0.9449**
	2 workers	0.8273	0.8273	0.8270
	4 workers	0.9235	0.9232	0.9233
	8 workers	0.9254	0.9254	**0.9252**
IoT-DNL	Single training	0.9961	0.9955	**0.9956**
	2 workers	0.9961	0.9955	**0.9956**
	4 workers	0.9917	0.9915	0.9917
	8 workers	0.9915	0.9913	0.9914

to different processes and communicate using Message Passing Interface (MPI). The results show that FL training is faster than single training in all cases, reducing the time to train the model while achieving similar performance by up to 89.3%. We can also see that 4 or 8 workers are the best options compared to 2 workers, as the results are better and the training time is lower.

TABLE IV: Training times until early stop

Dataset	Model	Time (s)	Time (%)
UNSW-NB15	Single training	40.58	100.0
	2 workers	19.82	48.8
	4 workers	16.54	40.7
	8 workers	17.59	43.3
ToN-IoT	Single training	417.37	100.0
	2 workers	53.76	12.8
	4 workers	44.96	10.7
	8 workers	54.95	13.1
IoT-DNL	Single training	36.88	100.0
	2 workers	20.48	55.5
	4 workers	9.10	24.6
	8 workers	9.91	24.6

This demonstrates the feasibility of utilizing the FL technique to train a model capable of detecting and classifying intrusions in IoT Networks without sharing the data with a central server, guaranteeing the privacy of the entities involved.

VIII. Conclusions and Future work

This paper delved into the challenges posed by securing IoT devices due to their limited resources and the continuous growth of IoT networks. The study focused on leveraging Machine Learning techniques, particularly Federated Learning, to develop effective IDS models while ensuring data privacy.

Through experiments conducted on three state-of-the-art datasets, namely UNSW-NB15, ToN-IoT, and IoT-DNL, the efficacy of FL in training IDS models was evaluated. The results demonstrated that FL-based training, with varying numbers of workers, yielded comparable performance to single training approaches across multiple performance metrics. Additionally, FL training exhibited faster convergence times compared to traditional single training methods, highlighting its efficiency and scalability.

These findings highlight the promise of FL for training IDS models in IoT networks, addressing the critical need for privacy-preserving techniques in data-driven security solutions. By decentralizing the training process and allowing devices to learn from distributed data collaboratively, FL offers a robust framework for enhancing network security while mitigating privacy concerns.

Although the results are promising, there are several limitations to this study. On the one hand, the experiments were simulated on a single machine, which means that the workers and the parameter server were running on the same hardware using different processes. This setup deviates from the real-world scenario where the workers and the parameter server would be running on different physical devices, potentially introducing communication costs and latency. On the other hand, the experiments were conducted only following a single FL approach with the dataset equally distributed over the workers, and there was no extensive hyperparameter tuning of specific FL parameters, including local and global epochs.

Considering the aforementioned limitations, future research avenues could further explore the communication costs associated with FL training since it can have a major impact and investigate the scalability of the approach with other FL approaches and larger datasets with non-IID distributions. Furthermore, the study could be extended to evaluate the performance of FL-based IDS models in adversarial settings, where attackers attempt to manipulate the training process to compromise the model's integrity.

Acknowledgements

This work is partially supported by the European Union / Next Generation EU, through Programa de Recuperação e Resiliência (PRR) [Project Nr. 29: Route 25 (02/C05-i01.01/2022.PC645463824-00000063)] and by FCT/MCTES through national funds and when applicable co-funded EU funds under the project UIDB/50008/2020-UIDP/50008/2020.

References

[1] P. Dini, A. Elhanashi, A. Begni, S. Saponara, Q. Zheng, and K. Gasmi, "Overview on intrusion detection systems design exploiting machine learning for networking cybersecurity," *Applied Sciences*, vol. 13, no. 13, 2023.

[2] T. Alkhalifah, H. Wang, and O. Ovcharenko, "Mlreal: Bridging the gap between training on synthetic data and real data applications in machine learning," 2021.

[3] Z. Li, V. Sharma, and S. P. Mohanty, "Preserving data privacy via federated learning: Challenges and solutions," *IEEE Consumer Electronics Magazine*, vol. 9, no. 3, pp. 8–16, 2020.

[4] S. M. Kasongo and Y. Sun, "A deep learning method with wrapper based feature extraction for wireless intrusion detection system," *Computers & Security*, vol. 92, p. 101752, 2020.

[5] M. M. Hassan, A. Gumaei, A. Alsanad, M. Alrubaian, and G. Fortino, "A hybrid deep learning model for efficient intrusion detection in big data environment," *Information Sciences*, vol. 513, pp. 386–396, 2020.

[6] G. Liu, H. Zhao, F. Fan, G. Liu, Q. Xu, and S. Nazir, "An enhanced intrusion detection model based on improved knn in wsns," *Sensors*, vol. 22, no. 4, p. 1407, 2022.

[7] Y. Imrana, Y. Xiang, L. Ali, and Z. Abdul-Rauf, "A bidirectional lstm deep learning approach for intrusion detection," *Expert Systems with Applications*, vol. 185, p. 115524, 2021.

[8] K. Jiang, W. Wang, A. Wang, and H. Wu, "Network intrusion detection combined hybrid sampling with deep hierarchical network," *IEEE access*, vol. 8, pp. 32464–32476, 2020.

[9] T.-T.-H. Le, Y. E. Oktian, and H. Kim, "Xgboost for imbalanced multiclass classification-based industrial internet of things intrusion detection systems," *Sustainability*, vol. 14, no. 14, p. 8707, 2022.

[10] M. A. Khan, "Hcrnnids: Hybrid convolutional recurrent neural network-based network intrusion detection system," *Processes*, vol. 9, no. 5, p. 834, 2021.

[11] T. Saba, A. Rehman, T. Sadad, H. Kolivand, and S. A. Bahaj, "Anomaly-based intrusion detection system for iot networks through deep learning model," *Computers and Electrical Engineering*, vol. 99, p. 107810, 2022.

[12] A. K. Balyan, S. Ahuja, U. K. Lilhore, S. K. Sharma, P. Manoharan, A. D. Algarni, H. Elmannai, and K. Raahemifar, "A hybrid intrusion detection model using ega-pso and improved random forest method," *Sensors*, vol. 22, no. 16, p. 5986, 2022.

[13] J. Gu and S. Lu, "An effective intrusion detection approach using svm with naïve bayes feature embedding," *Computers & Security*, vol. 103, p. 102158, 2021.

[14] I. Parsa, "KDD Cup 1998 Data." UCI Machine Learning Repository, 1998. DOI: https://doi.org/10.24432/C5401H.

[15] S. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. Chan, "KDD Cup 1999 Data." UCI Machine Learning Repository, 1999. DOI: https://doi.org/10.24432/C51C7N.

[16] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pp. 1–6, 2009.

[17] K. Kostas, M. Just, and M. A. Lones, "Iotgem: Generalizable models for behaviour-based iot attack detection," *arXiv preprint arXiv:2401.01343*, 2023.

[18] 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, 2015.

[19] T. M. Booij, I. Chiscop, E. Meeuwissen, N. Moustafa, and F. T. H. d. Hartog, "Ton_iot: The role of heterogeneity and the need for standardization of features and attack types in iot network intrusion data sets," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 485–496, 2022.

[20] M. Sarhan, S. Layeghy, and M. Portmann, "Towards a standard feature set for network intrusion detection system datasets," *Mobile Networks and Applications*, vol. 27, p. 357–370, Nov. 2021.

[21] R. Abou Khamis and A. Matrawy, "Evaluation of adversarial training on different types of neural networks in deep learning-based idss," in *2020 international symposium on networks, computers and communications (ISNCC)*, pp. 1–6, IEEE, 2020.

[22] P. Kumar, R. Tripathi, and G. P. Gupta, "P2idf: A privacy-preserving based intrusion detection framework for software defined internet of things-fog (sdiot-fog)," in *Adjunct Proceedings of the 2021 International Conference on Distributed Computing and Networking*, pp. 37–42, 2021.

[23] D. K. K. Reddy, J. Nayak, B. Naik, and G. S. Pratyusha, "11 deep neural network–based security model for iot device network," *Deep Learning for Internet of Things Infrastructure*, p. 223, 2021.

[24] M. Langer, Z. He, W. Rahayu, and Y. Xue, "Distributed training of deep learning models: A taxonomic perspective," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, p. 2802–2818, Dec. 2020.

[25] R. Teixeira, M. Antunes, D. Gomes, and R. Aguiar, "The learning costs of federated learning in constrained scenarios," 08 2023.

[26] M. Sarhan, S. Layeghy, and M. Portmann, "Feature analysis for machine learning-based iot intrusion detection," *arXiv preprint arXiv:2108.12732*, 2021.