

# Optimizing Federated Learning for Intrusion Detection in IoT Networks

Abderahmane Hamdouchi<sup>a</sup> and Ali Idri<sup>b</sup>  
Mohammed VI Polytechnic University, Benguerir, Morocco

**Keywords:** Intrusion Detection System, Federated Learning, Deep Learning, Netflow, IoT, Cybersecurity.

**Abstract:** The Internet of Things (IoT) involves billions of interconnected devices, making IoT networks vulnerable to cyber threats. To enhance security, deep learning (DL) techniques are increasingly used in intrusion detection systems (IDS). However, centralized DL-based IDSs raise privacy concerns, prompting interest in Federated Learning (FL). This research evaluates FL configurations using dense neural networks (DNN) and convolutional neural networks (CNN) with two optimizers, stochastic gradient descent (SGD) and Adam, across 20% and 60% feature thresholds. Two cost-sensitive learning techniques were applied: undersampling with binary cross-entropy and weighted classes using weighted binary cross-entropy. Using the NF-ToN-IoT-v2 dataset, 16 FL configurations were analyzed. Results indicate that SGD, combined with CNN and the Undersampling technique applied to the top 7 features, outperformed other configurations.


## 1 INTRODUCTION


In the current digital landscape, the volume of data generated and stored has surged dramatically, driven by the decreasing cost of storage and increasing tendency to record every digital interaction. This data proliferation is further exacerbated by the growing adoption of Internet of Things (IoT) devices, including smart home technologies, the expansion of smart cities, and advancements associated with Industry 4.0. For big data companies, insights derived from IoT devices are invaluable, rendering data a critical asset that requires robust protection (Atharvan et al., 2022).

Intrusion detection systems (IDS) are essential in securing IoT environments. The integration of machine learning (ML) into an IDS enhances threat detection capabilities, but ML models often encounter challenges when dealing with imbalanced data, leading to complications in model design and an increased risk of overfitting (de Zarzà et al., 2023; Thakkar & Lohiya, 2023). Several strategies have been investigated to address these challenges: (1) cost-sensitive learning, which includes techniques such as cost-sensitive resampling (oversampling and undersampling (Luengo et al., 2011)) to adjust data

distribution; (2) cost-sensitive algorithms, where ML algorithms are modified to incorporate a cost matrix, although this approach is time-consuming (Lomax & Vadera, 2013); (3) cost-sensitive ensembles, which combine predictions from traditional ML models while accounting for misclassification costs (Krawczyk et al., 2014; Tao et al., 2019); and (4) cost-sensitive learning in deep learning (DL), where model training involves adjusting weights using standard loss functions such as weighted binary cross entropy (WBCE) (Ho & Wookey, 2020), specifically developed to address the challenges posed by imbalanced data (Dina et al., 2023; Kerkhof et al., 2022).

However, early IDS systems were hindered by their lack of adaptability and slow response times to emerging threats, leaving them exposed for prolonged periods (Murphy, 2018). To address these limitations, more advanced IDS systems have begun to employ basic ML models to autonomously learn and identify new threats. Although the incorporation of ML has improved the accuracy of attack detection, most ML-based IDS systems remain centralized. This centralization involves a single organization collecting and processing data from multiple devices to train its ML models, raising significant privacy

<sup>a</sup>  <https://orcid.org/0009-0002-6623-8276>

<sup>b</sup>  <https://orcid.org/0000-0002-4586-4158>

concerns. This is particularly relevant in IoT environments such as smart wearables and healthcare devices, where sensitive and large volumes of data are at stake. Consequently, there is a growing demand for decentralized approaches to data management (McMahan et al., 2017).

To address the privacy concerns inherent in centralized ML approaches, federated learning (FL) (McMahan et al., 2017) was introduced in 2016. FL allows multiple devices, often referred to as clients or parties, to collaboratively train a model without sharing their data directly. Instead, these devices send model updates to a central entity known as an aggregator or coordinator, where the updates are combined to refine a global model. This approach is designed to enhance privacy by ensuring that the data remains local to the devices, thereby reducing the risk of exposure. FL achieves this by transmitting only the gradients instead of the raw data itself. The local training occurs on each device using its own dataset, and only the computed model parameters are sent to the central server. As a result, sensitive information never leaves the device, significantly reducing the risk of interception or unauthorized access during transmission.

Recent efforts have focused on developing FL-based IDS for IoT environments (Hei et al., 2020; Nguyen et al., 2019; Thu Huong et al., 2020). Despite this progress, many proposed approaches do not adequately address the challenges of imbalanced data, leading to models that are prone to overfitting. Additionally, these methods often fail to evaluate the effectiveness of different FL optimizers and DL base learners, or consider the impact of varying feature sets. Furthermore, the review (Agrawal et al., 2021) highlighted the challenges of implementing FL in IoT settings, but failed to provide concrete recommendations for enhancing IDS with FL or critically assessing the proposed solutions. This gap in detailed analysis poses challenges for cybersecurity experts in identifying the key issues associated with integrating FL into an IDS for IoT.

To address the existing research gap, this study investigates 16 FL configurations for IDS in IoT contexts, specifically employing deep learning (DL) through dense neural networks (DNN) and convolutional neural networks (CNN). These configurations (16 = 2 optimizers for the FL server  $\times$  2 DL architectures  $\times$  2 cost-sensitive configurations  $\times$  2 feature thresholds) explore the impact of different cost-sensitive learning approaches (resampling based on WBCE) along with the selection of FL optimizers and the number of features. The study utilized the NF-

ToN-IoT-v2 dataset to evaluate the effectiveness of FL with various cost-sensitive setups, feature numbers, and FL optimizers. We conduct 16 experiments: eight with the dataset's original distribution using WBCE, and eight with undersampled data to balance attack and non-attack instances among participants using binary cross entropy (BCE). The study assesses the results using two FL optimizers: stochastic gradient descent (SGD) (Amari, 1993) and Adam (Zhang, 2019), with two feature thresholds (20% and 60%). The performance of the FL models was evaluated in binary classification tasks over 100 optimization rounds using four performance criteria: accuracy, AUC, precision, and recall. The Scott–Knott (SK) statistical test (Scott & Knott, 1974) and the Borda count (BC) voting system (Saari, 2001) were employed to compare and rank the models.

This study addresses the following research questions (RQs):

- **(RQ1).** What is the optimal choice between SGD and Adam optimizers in the context of FL for attack detection?
- **(RQ2).** What is the best FL configuration for detecting attacks across various settings?

The key contributions of this study are as follows:

1. Determining the best optimizer for FL in the context of intrusion detection using the NF-ToN-IoT-v2 dataset.
2. Identifying optimal FL setup for different configurations.
3. Developing a generalized FL model for IDS applicable to various IoT datasets.

The remainder of this paper is organized as follows. Section 2 provides a review of related literature. Section 3 presents the data used in this study. Section 4 outlines the research methodology. Section 5 presents the results and discussion of the experiments. Finally, Section 6 presents the conclusions and suggests directions for future research.

## 2 RELATED WORK

Several significant studies have explored anomaly detection across various domains, with a particular emphasis on IoT, utilizing different FL methodologies. This section provides an overview of key research efforts that have employed FL for intrusion detection within IoT environments.

Friha et al. (Friha et al., 2022) proposed an FL-based IDS (FELIDS) aimed at securing agricultural IoT infrastructure by ensuring data privacy through

localized learning. To defend against attacks on Agricultural IoT systems, FELIDS leverages three DL classifiers: DNN, CNN, and recurrent neural networks (RNN). The effectiveness of the system was evaluated using three datasets: CSE-CIC-IDS2018, MQTTset, and InSDN. The findings indicate that FELIDS outperforms traditional centralized machine learning (non-FL) methods by offering enhanced data privacy for IoT devices and achieving an accuracy of 99.71% with the CNN classifier on the InSDN dataset, 89.56% with the RNN classifier on the MQTTset dataset, and 94.15% with the RNN classifier on the CSE-CIC-IDS2018 dataset. Mothukuri et al. (Mothukuri et al., 2022) proposed a novel approach that leverages FL to train gated recurrent units (GRUs) models while ensuring that data remains on local IoT devices. Only the learned weights of the model are shared with the central server of the FL. In addition, the method incorporates an ensemble technique to aggregate updates from multiple sources, thereby improving the accuracy of the global ML model. The results demonstrated that this approach not only outperforms traditional centralized ML methods in safeguarding data privacy but also achieves an overall average accuracy of 90.255% in attack detection. Idrissi et al. (Idrissi et al., 2023) introduced Fed-ANIDS, a Network IDS that combines ML-based anomaly detection (AD) with FL to address privacy concerns inherent in centralized models. The system detects intrusions by calculating an intrusion score based on the reconstruction error of normal traffic using various AD models, including simple autoencoders, variational autoencoders, and adversarial autoencoders. The method was evaluated using three datasets: USTC-TFC2016, CIC-IDS2017, and CSE-CIC-IDS2018. The results indicated that autoencoder-based models outperformed other generative adversarial network-based models, and that the FedProx aggregation framework was more effective than FedAVG. The proposed method achieved peak accuracies of 99.95%, 93.54%, and 94.48% for the USTC-TFC2016, CIC-IDS2017, and CSE-CIC-IDS2018 datasets, respectively.

### 3 EXPERIMENTAL DESIGN

This section describes the datasets, performance metrics, and methodology employed in the empirical evaluations conducted in this study.

#### 3.1 Dataset Description

In this study, 43 NetFlow features conforming to version 9 standard were extracted and labeled from the ToN-IoT (Alsaedi et al., 2020) dataset using the nProbe tool by Ntop. The resulting dataset, designated as NF-ToN-IoT-V2 (Sarhan et al., 2021), comprises 16,940,469 instances labeled as either attack or no-attack, reflecting a significant class imbalance, with 36% labeled as no-attack and 64% as attack. This data was chosen because it consists of real IoT traffic with simulated attacks. Furthermore, the "Pcap" files are labeled and available as open source, providing a foundation for developing a generalized model applicable to all NetFlow V9 data types.

#### 3.2 Performance Measures

To evaluate the effectiveness of the proposed binary classification models, we employed accuracy, recall, precision, and area under the receiver operating characteristic curve (AUC) as evaluation metrics (Naidu et al., 2023). These metrics were selected owing to their widespread use and acceptance in the field.

#### 3.3 Statistical Test and Borda Count

- **Scott Knott (SK)** is a clustering algorithm commonly used to compare multiple groups in analysis of variance studies. It addresses the issue of overlapping groups by starting with all observed mean effects grouped together and iteratively dividing these groups into smaller subgroups, ensuring that no two subgroups share any common members (Scott & Knott, 1974).
- **Borda Count (BC)** is a voting method in which voters rank candidates according to their preferences. Each candidate receives points based on their rank, with the lower ranks earning fewer points. The points are then aggregated, and the candidate with the highest total is declared as the winner. In this study, the Borda count method was used to identify the top-performing model, treating all performance measures equally (Saari, 2001).

#### 3.4 Methodology

Figure 1 illustrates the methodology employed to assess and compare the effects of different FL optimizers, DL architectures, cost-sensitive learning

setups, and feature thresholds on the detection capabilities of FL-based IDS. We evaluated the performance of two FL optimizers, SGD and Adam, with two different feature thresholds (20% and 60%) and two cost-sensitive learning setups (undersampling and WBCE) over 100 rounds using SK and BC voting systems. The experimental procedure included the following steps:

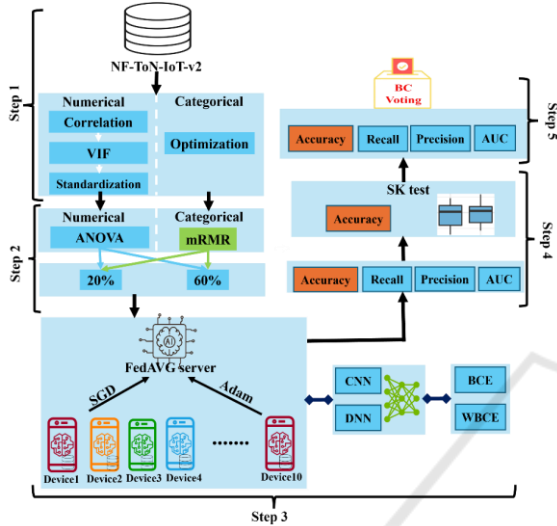


Figure 1: Experimental process.

- **Step 1.** Prepare the raw data by removing missing values, duplicate rows, and unnecessary attributes. This process includes reclassifying categorical features and standardizing numerical features.
- **Step 2.** involves the application of two FS techniques: mRMR for categorical features and ANOVA for numerical features. These methods were selected based on a comparative analysis of FS and ML techniques on IDS datasets (Amiri et al., 2011; Shakeela et al., 2021; Tao et al., 2019; Zouhri et al., 2023). Additionally, two feature thresholds (20% and 100%) were implemented, as recommended in previous studies (Dhaliwal et al., 2018; Kurniabudi et al., 2020; Nakashima et al., 2018). This process resulted in the creation of two dataset variations.
- **Step 3.** Set up a simulated IoT network by creating virtual instances using TFF based on the DNN and CNN architectures. We utilized 10 devices, each referred to as  $Device_i$ , and configured two optimizers for the central FedAVG server instance. This instance facilitates the exchange of DL model parameters between the mobile IoT devices and

the central FL server. Cost-sensitive learning was implemented using two approaches: one using raw data with WBCE, respecting the original dataset distribution, and the other using undersampling to balance the dataset with equal numbers of attacks and BCE. Each local dataset  $i$  was assigned to the corresponding virtual  $Device_i$ .

- **Step 4.** Construct and evaluate the performance of the 16 FL configurations ( $16 = 2$  optimizers for the FL server  $\times 2$  DL architectures  $\times 2$  cost-sensitive configurations  $\times 2$  feature thresholds) in terms of accuracy, recall, precision, and AUC over 100 rounds. Additionally, the SK test and BC system were used to rank the FL configurations for each cost-sensitive learning setup, feature threshold, and FL optimizer.
- **Step 5.** Compare the performances of Adam and SGD optimizers for each cost-sensitive learning setup and feature threshold using the NF-ToN-IoT-v2 dataset. Ultimately, identify the optimal FL configuration for cyber-detection within the NetFlow IoT dataset framework.

### 3.5 Abbreviation

To enhance readability and simplify model names, this study adopts the following specific naming conventions:

*DLArchitecture\_Optimizer\_CostSensitiveSetup\_FeatureThreshold*

The abbreviations for DL architectures are DN for DNN and CN for CNN. The FL optimizers are abbreviated as S for SGD and A for Adam. The cost-sensitive setups are abbreviated as U for undersampling and W for weighted classes. For instance, DNSW20 represents a configuration utilizing the DNN architecture with the SGD optimizer, WBCE with weighted classes, and 20% of the features.

## 4 RESULTS AND DISCUSSION

This section analyzes the outcomes of using the FL technique with the DNN and CNN architectures. The evaluation includes two optimizers (Adam and SGD), two feature thresholds (20% and 60%), and two cost-sensitive setups (undersampling and WBCE) over 100 rounds on the NF-ToN-IoT-v2 dataset for binary classification. The results of the empirical study are discussed in relation to the RQs introduced in section 1.

### 4.1 Optimal Choice Between SGD and Adam Optimizers in FL for Attack Detection (RQ1)

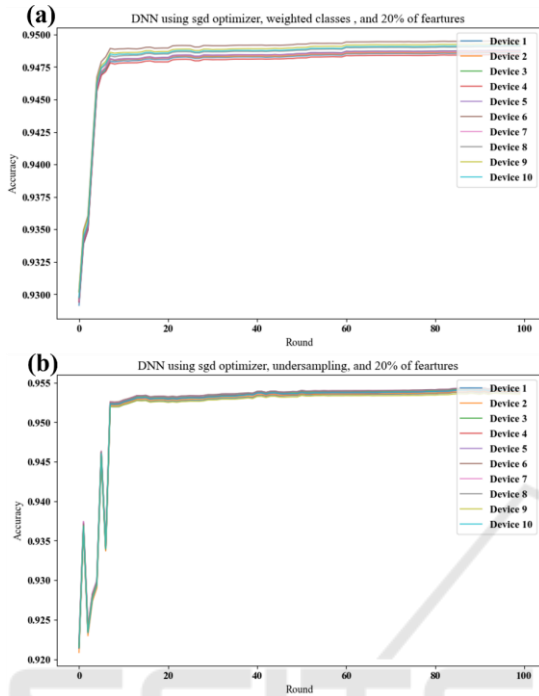


Figure 2: Accuracy progression across rounds for 10 devices using SGD with 20% of features: (a) DNN with weighted classes and (b) DNN with undersampling.

This subsection investigates the impact of SGD and Adam optimizers on the performance of FL configurations, with an emphasis on identifying the optimizer that enhances the accuracy of an FL-based IDS in IoT contexts. We analyze the average model accuracy across different feature thresholds for each cost-sensitive learning setup using DNN and CNN architectures, utilizing the SGD and Adam optimizers over 100 rounds to determine the optimal FL optimizer. For instance, when evaluating the accuracy of cost-sensitive setups (undersampling and weighted classes) with SGD and Adam using DNN and 20% of features across 10 devices: (1) Figures 2.a and 2.b display the accuracy values of DNN using the SGD optimizer with weighted classes and undersampling, respectively, allowing us to assess the models' accuracy for each device over 100 rounds; and (2) we calculate the average accuracy values for the 10 devices for different FL configurations in each round, as illustrated in Figure 3. For example, as shown in Figure 3. a, the average accuracy of the FL architecture across 10 devices, deploying a DNN with weighted classes and utilizing 20% of features with

SGD as the FL optimizer, is referred to as DNSW20, whereas the average accuracy of the FL architecture across 10 devices, deploying a DNN with undersampling and using 20% of features with SGD as the FL optimizer, is indicated as DNSU20.

Figure 3.a shows the average accuracy values obtained using the DNN with 20% of the features. We observe the following:

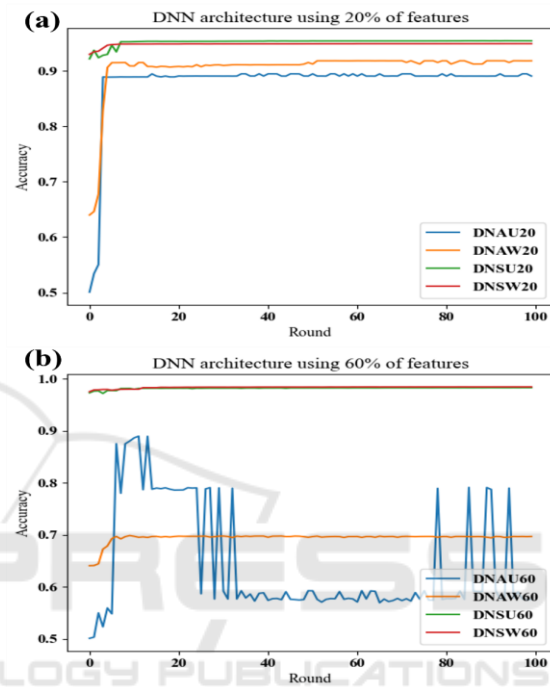


Figure 3: Average accuracy of FL based on DNN architecture using: (a) 20% of features and (b) 60% of features.

- For DNSU20, variability is present in the first 5 rounds, followed by stabilization, achieving a high accuracy of approximately 95%.
- For DNSW20, variability is present in the first 3 rounds, followed by stabilization, resulting in a high accuracy of approximately 94%.
- For DNAW20, an increase is observed in the first 3 rounds, followed by stabilization with very slight variations, leading to a consistent accuracy of approximately 91%.
- For DNAU20, an increase is observed in the first 5 rounds, followed by stabilization with very slight variations, resulting in a consistent accuracy of approximately 89%.

Figure 3.b shows the average accuracy values obtained using the DNN with 60% of the features. We observe the following:

- For DNSW60, stability is maintained throughout all rounds, achieving a high accuracy of approximately 98%.
- For DNSU60, stability is maintained in all rounds except for a slight variation in the second round, leading to a high accuracy of approximately 98%.
- For DNAW60, an increase is observed in the first 6 rounds, followed by stabilization, resulting in a consistent accuracy of approximately 91%.
- For DNAU60, the highest variation is observed across all rounds, with an initial increase in the first 5 rounds, some stability between rounds 5 and 15, followed by a decrease at round 36, and high variation at round 80, resulting in a maximum accuracy of 87%.

When using DNN as the base learner, the SGD optimizer proved to be the most effective FL optimizer, securing the top two positions for both the 20% and 60% feature thresholds. Specifically, DNSU20 for 20% of features and DNSW60 for 60% of features achieved the highest accuracies of 95% and 98%, respectively. In contrast, the Adam optimizer ranked lowest, with DNAU20 for 20% of features and DNAU60 for 60% of features, recording the lowest accuracies of 89% and 87%, respectively.

Figure 4.a illustrates the average accuracy values obtained using CNN with 20% of features. We observe the following:

- For CNSU20, a slight increase is observed during the first 5 rounds, followed by stabilization over the next 10 rounds. Variability occurs in the subsequent 3 rounds, after which stabilization occurs in the remaining rounds, achieving a high accuracy of approximately 95%.
- For CNSW20, a slight increase is observed during the first 5 rounds, followed by stabilization, ultimately leading to a high accuracy of approximately 94%.
- For CNAW20, a significant increase is observed during the first 3 rounds, followed by stabilization, resulting in a high accuracy of approximately 91%.
- For CNAU20, stabilization is observed throughout all rounds, resulting in a consistent accuracy of approximately 50%.

Figure 4.b shows the average accuracy values obtained using the CNN with 60% of the features. We observe the following:

- For CNSW60, a slight increase is observed during the first 5 rounds, followed by

stabilization, ultimately achieving a high accuracy of approximately 98%.

- For CNSU60, a significant increase is observed during the first 5 rounds, followed by stabilization, ultimately resulting in a high accuracy of approximately 98%.
- For CNAW60, a significant increase is observed during the first 10 rounds, followed by stabilization over the next 10 rounds. Significant variability is present in the remaining rounds, with accuracies ranging between 70% and 92%.
- For CNAU60, stabilization is observed throughout all rounds, resulting in a consistent accuracy of approximately 50%.

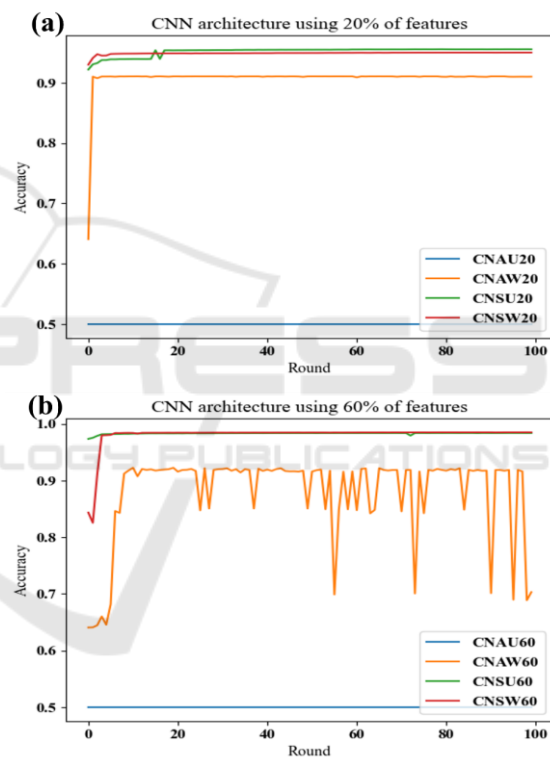


Figure 4: Average accuracy of FL based on CNN architecture using: (a) 20% of features and (b) 60% of features.

When using the CNN as the base learner, the SGD optimizer proved to be the most effective FL optimizer, securing the top two positions for both the 20% and 60% feature thresholds. Specifically, CNSU20 for 20% of features and CNSW60 for 60% of features achieved the highest accuracies of 95% and 98%, respectively. Conversely, the Adam optimizer ranked lowest, with CNAU20 stabilizing at an accuracy of 50% for 20% of the features and

CNAU60 showing variability with accuracy ranging between 70% and 92% for 60% of the features.

In summary, SGD outperformed Adam as an FL optimizer across DL architectures used as the base learners. Additionally, when utilizing SGD, employing weighted classes, particularly WBCE, as a cost-sensitive learning method yielded better performance with 60% of the features, while undersampling performed more effectively with 20% of the features across different DL architectures. On the other hand, when using Adam, the weighted classes (WBCE) consistently achieved better performance than the undersampling technique.

### 4.2 Optimal FL Configuration for Attack Detection Across Varied Settings (RQ2)

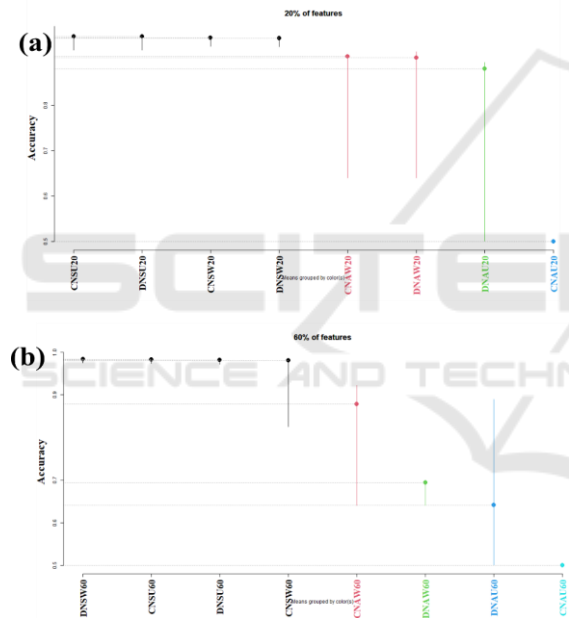


Figure 5: SK test results of FL configurations using (a) 20% of features and (b) 60% of features.

In this section, we compare various FL configurations, including FL optimizers, DL base learners, and cost-sensitive setups, for each feature threshold. The SK test was employed to focus on accuracy, grouping models, and identifying the most effective SK clusters, as illustrated in Figure 5. Additionally, the BC method was used to prioritize the models within the top SK clusters based on metrics such as accuracy, AUC, recall, and precision, as shown in Table 2. The SK test results are displayed in a graph, where the x-axis categorizes the FL classifier variants by cluster, arranging the best

clusters from left to right, and the y-axis shows the accuracy scores. The central dots on each vertical line represent the mean accuracy, with the lines illustrating the outcomes of 100 rounds for each FL classifier. This analysis involved calculating the average accuracy for each round  $i$  across the 10 devices, denoted as  $Average_i$ , using Equation (1). For example, DNSW20 represents the average accuracy of the DNN architecture with the SGD optimizer, WBCE with weighted classes, and 20% of the features across 10 over 100 rounds ( $Average_1, \dots, Average_{100}$ ).

$$Average_i = \sum_j^{\#Device} \frac{Accuracy_j}{\#Devices} \tag{1}$$

From Figure 5.a, we observe the following:

- For the 20% feature threshold, the SK distribution results in four distinct clusters. The first cluster comprises all models utilizing the SGD optimizer, including CNSU20, DNSU20, CNSW20, and DNSW20. The second cluster consists of CNAW20 and DNAW20. The third cluster includes only DNAU20, while the fourth cluster contains only CNAU20.
- For the 60% feature threshold, the SK distribution forms five distinct clusters. The first cluster includes all models using the SGD optimizer, specifically DNSW60, CNSU60, DNSU60, and CNSW60. The second, third, fourth, and fifth clusters contain CNAW60, DNAW60, DNAU60, and CNAU60, respectively.

Table 1: BC ranking of the FL variants belong to the best SK cluster.

TS	#F	Model	Accuracy	AUC	Recall	Precision	BC
20%	7	CNSU20	95.19%	0.9874	92.41%	97.85%	9
		CNSW20	94.89%	0.9872	97.33%	94.81%	8
		DNSU20	95.18%	0.9859	92.61%	97.62%	7
		DNSW20	94.78%	0.9859	97.26%	94.73%	6
60%	20	DNSW60	98.34%	0.9984	99.03%	98.38%	11
		DNSU60	98.17%	0.9984	98.52%	97.84%	7
		CNSU60	98.31%	0.9987	98.23%	98.39%	7
		CNSW60	98.07%	0.9943	98.48%	98.49%	5

TS: Feature threshold #F: Number of features

The findings indicate that the optimal FL configuration is achieved using the SGD optimizer and varies based on the feature threshold. Specifically, (1) with 20% of the features, the CNN base learner combined with the SGD optimizer outperforms the DNN, securing the top two positions

according to the BC voting system. Conversely, (2) with 60% of the features, the DNN occupies the top two positions. Specifically, the best model for the 20% feature threshold is CNSU20, which achieved the highest BC score of 9 with an accuracy of 95.19%, an AUC of 0.9874, a recall of 92.41%, and a precision of 97.85% using 7 features. For the 60% feature threshold, DNSW60 achieved the best BC score of 11 with an accuracy of 98.34%, an AUC of 0.9984, a recall of 99.03%, and a precision of 98.38% using 20 features. Although CNSU20 and DNSW60 demonstrated comparable performance across various metrics, CNSU20 was selected as the preferred model due to its effectiveness with a minimal number of features (7 features).

## 5 CONCLUSION AND FURTHER WORKS

The research evaluated and compared 16 FL configurations for the binary classification of network intrusions, employing DNN and CNN as base learning models. The study explored the performance of two FL optimizers, SGD and Adam, in combination with two feature thresholds (20% and 60%) and two cost-sensitive learning approaches (Undersampling with BCE and weighted classes with WBCE) using the NF-ToN-IoT-v2 dataset. Evaluation metrics included accuracy, AUC, recall, and precision, further supported by the SK statistical test and the BC ranking system. The results demonstrated that SGD is a more reliable optimizer for attack detection in FL frameworks. The most effective model configuration was achieved using SGD as the FL optimizer, combined with CNN as the base learner and the Undersampling technique over the top 7 features.

The findings underscore the significance of employing FL in the development of decentralized IDSs specifically tailored for IoT networks to enhance attack detection. Future research should extend empirical evaluations to further validate or refine these results, potentially by utilizing a variety of datasets to assess the robustness and adaptability of FL-based IDS across diverse IoT environments. Additionally, investigating alternative models within FL frameworks could offer valuable insights into optimizing both performance and efficiency. Furthermore, deploying these models on embedded devices using TinyML and FL methodologies represents a promising direction for continued exploration.

## REFERENCES

- Agrawal, S., Sarkar, S., Aouedi, O., Yenduri, G., Piamrat, K., Alazab, M., Bhattacharya, S., Maddikunta, P. K. R. & Gadekallu, T. R. (2021). Federated Learning for Intrusion Detection System: Concepts, Challenges and Future Directions. *Computer Communications*, 195, 346–361. <https://doi.org/10.1016/j.comcom.2022.09.012>
- Alsaedi, A., Moustafa, N., Tari, Z., Mahmood, A. & Adna N Anwar. (2020). TON-IoT telemetry dataset: A new generation dataset of IoT and IIoT for data-driven intrusion detection systems. *IEEE Access*, 8, 165130–165150. <https://doi.org/10.1109/ACCESS.2020.3022862>
- Amari, S. ichi. (1993). Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4–5), 185–196. [https://doi.org/10.1016/0925-2312\(93\)90006-O](https://doi.org/10.1016/0925-2312(93)90006-O)
- Amiri, F., Rezaei Yousefi, M., Lucas, C., Shakery, A. & Yazdani, N. (2011). Mutual information-based feature selection for intrusion detection systems. *Journal of Network and Computer Applications*, 34(4), 1184–1199. <https://doi.org/10.1016/J.JNCA.2011.01.002>
- Atharvan, G., Koolikkara Madom Krishnamoorthy, S., Dua, A. & Gupta, S. (2022). A way forward towards a technology-driven development of industry 4.0 using big data analytics in 5G-enabled IIoT. *International Journal of Communication Systems*, 35(1), e5014. <https://doi.org/10.1002/DAC.5014>
- de Zarzà, I., de Curtò, J. & Calafate, C. T. (2023). Optimizing Neural Networks for Imbalanced Data. *Electronics* 2023, Vol. 12, Page 2674, 12(12), 2674. <https://doi.org/10.3390/ELECTRONICS12122674>
- Dhaliwal, S. S., Nahid, A. Al & Abbas, R. (2018). Effective Intrusion Detection System Using XGBoost. *Information* 2018, Vol. 9, Page 149, 9(7), 149. <https://doi.org/10.3390/INFO9070149>
- Dina, A. S., Siddique, A. B. & Manivannan, D. (2023). A deep learning approach for intrusion detection in Internet of Things using focal loss function. *Internet of Things*, 22, 100699. <https://doi.org/10.1016/J.IOT.2023.100699>
- Friha, O., Ferrag, M. A., Shu, L., Maglaras, L., Choo, K. K. R. & Nafaa, M. (2022). FELIDS: Federated learning-based intrusion detection system for agricultural Internet of Things. *Journal of Parallel and Distributed Computing*, 165, 17–31. <https://doi.org/10.1016/J.JPDC.2022.03.003>
- Hei, X., Yin, X., Wang, Y., Ren, J. & Zhu, L. (2020). A trusted feature aggregator federated learning for distributed malicious attack detection. *Computers & Security*, 99, 102033. <https://doi.org/10.1016/J.COSE.2020.102033>
- Ho, Y. & Wookey, S. (2020). The Real-World-Weight Cross-Entropy Loss Function: Modeling the Costs of Mislabeling. *IEEE Access*, 8, 4806–4813. <https://doi.org/10.1109/ACCESS.2019.2962617>
- Idrissi, M. J., Alami, H., El Mahdaouy, A., El Mekki, A., Oualil, S., Yartaoui, Z. & Berrada, I. (2023). Fed-ANIDS: Federated learning for anomaly-based network intrusion detection systems. *Expert Systems with*



- Applications, 234, 121000. <https://doi.org/10.1016/J.ESWA.2023.121000>
- Kerkhof, M., Wu, L., Perin, G. & Picek, S. (2022). Focus is Key to Success: A Focal Loss Function for Deep Learning-Based Side-Channel Analysis. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 13211 LNCS, 29–48. [https://doi.org/10.1007/978-3-030-99766-3\\_2/COVER](https://doi.org/10.1007/978-3-030-99766-3_2/COVER)
- Krawczyk, B., Woźniak, M. & Schaefer, G. (2014). Cost-sensitive decision tree ensembles for effective imbalanced classification. *Applied Soft Computing*, 14(PART C), 554–562. <https://doi.org/10.1016/J.ASOC.2013.08.014>
- Kurniabudi, Stiawan, D., Darmawijoyo, Bin Idris, M. Y. Bin, Bamhdi, A. M. & Budiarto, R. (2020). CICIDS-2017 Dataset Feature Analysis with Information Gain for Anomaly Detection. *IEEE Access*, 8, 132911–132921. <https://doi.org/10.1109/ACCESS.2020.3009843>
- Lomax, S. & Vadera, S. (2013). A survey of cost-sensitive decision tree induction algorithms. *ACM Computing Surveys (CSUR)*, 45(2). <https://doi.org/10.1145/2431211.2431215>
- Luengo, J., Fernández, A., García, S. & Herrera, F. (2011). Addressing data complexity for imbalanced data sets: Analysis of SMOTE-based oversampling and evolutionary undersampling. *Soft Computing*, 15(10), 1909–1936. <https://doi.org/10.1007/S00500-010-0625-8/TABLES/16>
- McMahan, B., Moore, E., Ramage, D., Hampson, S. & Arcas, B. A. y. (2017). Communication-Efficient Learning of Deep Networks from Decentralized Data (pp. 1273–1282). *PMLR*. <https://proceedings.mlr.press/v54/mcmahan17a.html>
- Mothukuri, V., Khare, P., Parizi, R. M., Pouriyeh, S., Dehghantanha, A. & Srivastava, G. (2022). Federated-Learning-Based Anomaly Detection for IoT Security Attacks. *IEEE Internet of Things Journal*, 9(4), 2545–2554. <https://doi.org/10.1109/JIOT.2021.3077803>
- Murphy, J. F. A. (2018). The General Data Protection Regulation (GDPR). *Irish Medical Journal*, 111(5), 747. <https://doi.org/10.1007/978-3-319-57959-7>
- Naidu, G., Zuva, T. & Sibanda, E. M. (2023). A Review of Evaluation Metrics in Machine Learning Algorithms. *Lecture Notes in Networks and Systems*, 724 LNNS, 15–25. [https://doi.org/10.1007/978-3-031-35314-7\\_2/FIGURES/3](https://doi.org/10.1007/978-3-031-35314-7_2/FIGURES/3)
- Nakashima, M., Kim, Y., Kim, J., Kim, J. & Sim, A. (2018). Automated Feature Selection for Anomaly Detection in. *Network Traffic Data*, 1(1), 27. <https://doi.org/10.1145/1122445.1122456>
- Nguyen, T. D., Marchal, S., Miettinen, M., Fereidooni, H., Asokan, N. & Sadeghi, A. R. (2019). D<sup>2</sup>IoT: A federated self-learning anomaly detection system for IoT. *Proceedings - International Conference on Distributed Computing Systems*, 2019-July, 756–767. <https://doi.org/10.1109/ICDCS.2019.00080>
- Saari, D. G. (2001). Decisions and Elections. *Decisions and Elections*. <https://doi.org/10.1017/CBO9780511606076>
- Sarhan, M., Layeghy, S., Moustafa, N. & Portmann, M. (2021). NetFlow Datasets for Machine Learning-Based Network Intrusion Detection Systems. *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, 371 LNICST, 117–135. [https://doi.org/10.1007/978-3-030-72802-1\\_9/COVER](https://doi.org/10.1007/978-3-030-72802-1_9/COVER)
- Scott, A. J. & Knott, M. (1974). A Cluster Analysis Method for Grouping Means in the Analysis of Variance. *Biometrics*, 30(3), 507. <https://doi.org/10.2307/2529204>
- Shakeela, S., Sai Shankar, N., Mohan Reddy, P., Kavya Tulasi, T. & Mahesh Koneru, M. (2021). Optimal Ensemble Learning Based on Distinctive Feature Selection by Univariate ANOVA-F Statistics for IDS. <https://doi.org/10.24425/ijet.2021.135975>
- Tao, X., Li, Q., Guo, W., Ren, C., Li, C., Liu, R. & Zou, J. (2019). Self-adaptive cost weights-based support vector machine cost-sensitive ensemble for imbalanced data classification. *Information Sciences*, 487, 31–56. <https://doi.org/10.1016/J.INS.2019.02.062>
- Thakkar, A. & Lohiya, R. (2023). Attack Classification of Imbalanced Intrusion Data for IoT Network Using Ensemble-Learning-Based Deep Neural Network. *IEEE Internet of Things Journal*, 10(13), 11888–11895. <https://doi.org/10.1109/JIOT.2023.3244810>
- Thu Huong, T., Phuong Bac, T., Minh Long, D., Doan Thang, B., Duc Luong, T., Thanh Binh, N. & Kim Phuc, T. (2020). LockEdge: Low-Complexity Cyberattack Detection in IoT Edge Computing. <https://arxiv.org/abs/2011.14194v1>
- Zhang, Z. (2019). Improved Adam Optimizer for Deep Neural Networks. *2018 IEEE/ACM 26th International Symposium on Quality of Service, IWQoS 2018*. <https://doi.org/10.1109/IWQOS.2018.8624183>
- Zouhri, H., Idri, A. & Ratnani, A. (2023). Evaluating the impact of filter-based feature selection in intrusion detection systems. *International Journal of Information Security*, 1–27. <https://doi.org/10.1007/S10207-023-00767-Y/TABLES/17>