Received 23 September 2024, accepted 4 November 2024, date of publication 8 November 2024,
date of current version 26 November 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3494057

RESEARCH ARTICLE

Lightweight Federated Learning for Efficient Network Intrusion Detection

ABDELHAK BOUAYAD[1], HAMZA ALAMI[2], MERYEM JANATI IDRISSI[1], AND ISMAIL BERRADA[1]
[1]College of Computing, Mohammed VI Polytechnic University (UM6P), Ben Guerir 43150, Morocco
[2]Laboratory of Informatics, Signals, Automatics, and Cognitivism (LISAC), Faculty of Sciences Dhar El Mahraz, Sidi Mohammed Ben Abdellah University, Fes 30003, Morocco

Corresponding author: Abdelhak Bouayad (abdelhak.bouayad@um6p.ma)

ABSTRACT Network Intrusion Detection Systems (NIDS) play a crucial role in ensuring cybersecurity across various digital infrastructures. However, traditional NIDS face significant challenges, including high computational and storage costs, as well as privacy risks. To address these issues, we introduce a novel method called "Lightweight-Fed-NIDS," which harnesses federated learning and structured model pruning techniques for NIDS. The primary advantage of our contribution lies in the one-time computation of the pruning mask, without the need to access clients' data. This mask is then distributed to all clients and utilized to prune and optimize their local models. Furthermore, we leverage the power of Convolutional Neural Network (CNN) architectures, including ResNet-50, ResNet-101, and VGG-19, to extract essential features from raw traffic flows. We evaluate the performance of our method using various NIDS benchmark datasets, such as UNSW-NB15, USTC-TFC2016, and CIC-IDS-2017. Our technique achieves up to a 3X acceleration in training time compared to traditional, unpruned federated learning models, while maintaining a high detection rate of $\sim 99\%$. Additionally, our method reduces model size by 90%, demonstrating its efficiency and scalability for real-world NIDS deployments. These results highlight the potential of Lightweight-Fed-NIDS to enhance network security while addressing privacy concerns and resource constraints in distributed environments.

INDEX TERMS Network intrusion detection system, federated learning, pruning, deep learning.

I. INTRODUCTION

In today's interconnected world, network security has become a critical concern for organizations of all sizes. The increasing connectivity of various systems to the wider Internet has unintentionally turned them into appealing targets for malicious cyberattacks [1]. For instance,the attack known as BlackEnergy 3, which occurred in Ukraine in 2015, inflicted significant damage and disruption to the country's critical infrastructure. This attack targeted the Ukrainian power grid network, resulting in widespread power outages and affecting numerous regions. The attack employed various tactics, including malware propagation and remote access. Consequently, the demand for effective security solutions has intensified, with industries recognizing the indispensable nature of intrusion detection systems as a frontline defense mechanism [2]. Intrusion Detection Systems (IDS) have emerged as a prominent research area, garnering significant interest from academia and industry [3], [4]. IDS encompass a range of devices, software applications, or their combinations, designed to monitor network behaviors [1], [5]. Their primary objective is to detect malicious activities or policy violations by systematically collecting and analyzing all available network data (network traffic, system logs, etc.). It is essential to promptly report any detected malicious activity to system administrators, enabling appropriate remedial measures to be performed to mitigate any potential harm. On the one hand, IDS can be divided into two main types according to the detection method: misuse-based and anomaly-based. Misuse-based methods primarily rely on comparing the collected system information with known signatures stored in a misuse pattern database. This approach enables the effective identification of known intrusions [6]. However, they perform poorly when dealing

The associate editor coordinating the review of this manuscript and approving it for publication was Sawyer Duane Campbell.

with zero-day attacks (unknown attacks). Anomaly-based techniques compare the real-time behavior of a system with its defined normal behavior. When the deviation between the current behavior and the normal behavior exceeds a predefined threshold, an alert is triggered. These techniques are capable of detecting zero-day attacks (unknown attacks). On the other hand, IDS can be categorized into two categories acording to the system structure conforming to various data sources: network-based and host-based [6]. Network-based IDS (NIDS) employs network sensors to capture and analyze real-time network traffic data, utilizing data analysis techniques to identify malicious communications. Host-based IDS (HIDS) primarily focuses on monitoring the data specific to a particular host (logs, processes, etc.), enabling the identification of intrusion behaviors on the active node. In this work, we focus mainly on NIDS using anomaly detection techniques.

Modern networks are complex distributed systems comprising numerous components across multiple machines. These components collaborate by communicating and coordinating their actions to achieve synchronization and operate as a unified system. It is imperative to incorporate a fast, lightweight, and distributed machine learning (ML) technique, leveraging the power of distributed computing and ML algorithms, to detect intrusions with improved efficiency and effectiveness. Federated Learning (FL) [7] offers a distributed ML paradigm, allowing multiple entities or devices to collaboratively train a global model without sharing their local data. FL leverages entities or devices' computation and storage capabilities and exchanges only model parameters or updates with a central server.

From a technical point of view, various approaches have been proposed to tackle the problem of intrusion detection using ML. Moreover, deep learning-based (DL) models have been demonstrated to achieve superior performances when compared to shallow learning techniques [3], [8], [9], [10], [11], [12], [13]. However, these models are known for their computational demands; they rely on high-performance resources and extensive computing infrastructure. The training phase of deep learning models is particularly energy-intensive due to several factors, such as the complexity of the neural network model, and the volume and size of datasets. Managing and optimizing energy consumption when leveraging deep learning-based IDS models (both in the training and inference phases) is crucial to ensure efficient and sustainable operation in resource-constrained environments. Consequently, pruning techniques were introduced to eliminate redundant or insignificant parts of a neural network, such as weights, connections, neurons, or filters. Pruning can reduce the size, complexity, and energy consumption of deep learning models without compromising their performance [14], [15], [16], [17], [18].

In this paper, we introduce Lightweight-Fed-NIDS, a framework for network intrusion detection based on FL. It leverages the benefits of structured pruning and FL to achieve high detection performance and low computational complexity. First, the NIDS global server computes a data-free pruning mask without requiring any NIDS clients' data. This technique is applied only once at the start of training, making it a zero-shot pruning technique. Next, the NIDS global server disseminates the global model and pruning mask to NIDS clients, which then prune and optimize their local models. Finally, the global server constructs the final optimized global model by aggregating the optimized local models collected from NIDS clients. Our neural network models are based on CNN architectures, namely ResNet-50 [19], ResNet-101 [19], and VGG-19 [20], enabling the extraction of valuable features from raw traffic flows. To show the effectiveness of our technique, we performed extensive experiments with three known NIDS datasets, including CIC-IDS-2017 [21], UNSW-NB15 [22], and USTC-TFC2016 [23]. The obtained results show that the combination of FL, structured pruning, and deep learning-based detection techniques enables our method to achieve both detection effectiveness and lightweight computing efficiency.

To summarize, the main contributions of the paper are as follows:

- We design a lightweight and fast-learning NIDS machine-learning model architecture that can be trained and deployed on edge devices with limited computing capacity.
- We exploit a zero-shot pruning method that is data-independent and reduces the model's size and complexity without compromising the detection performance.
- We leverage the power of FL to enable collaborative NIDS model training and updating among multiple participants without sharing their private data.
- We assess the performance of our approach on various benchmark NIDS datasets and demonstrate its efficiency in terms of detection performance, training time complexity, and inference time complexity.

The rest of the paper is organized as follows. Section II reviews the related work on the topic. Section III introduces our proposed method and explains how it addresses the research problem. In Section IV, we evaluate our method on different datasets; Finally, in Section V, we conclude the paper and discuss some future works.

II. RELATED WORK

Our work lies at the intersection of three main fields: NIDS, FL, and deep learning pruning methods. In the following paragraphs, we describe relevant works within FL for NIDS and deep learning pruning techniques that are related to our work.

A. FEDERATED LEARNING FOR NIDS

In their work, Zhao et al. [9] proposed a multi-task deep neural network classifier in an FL system for network anomaly detection. The authors evaluated their method on three datasets: CIC-IDS-2017 [21], ISCXVPN2016 [24], and ISCXTor2016 [25]. They conducted two experiments with different combinations of datasets. The first experiment with CIC-IDS-2017 and ISCXVPN2016 achieved a detection accuracy of 98.14%, while the second experiment using CIC-IDS-2017 and ISCXTor2016 dropped to 97.81%.

For cyberattacks targeting industrial Cyber-Physical Systems (CPSs), Li et al. [2] proposed a federated deep learning approach called DeepFed. They developed a CNN-GRU-based intrusion detection model and created a framework for multiple industrial CPSs to collaborate on model development. The anonymity of model parameters is protected using a secure communication protocol based on the Paillier cryptosystem. Performance evaluation on a real-world dataset from a gas pipelining system demonstrates that DeepFed achieved high accuracy, precision, recall, and F1-score, with all measures exceeding 97% for various numbers of local agents.

A novel algorithm for Asynchronous Federated Learning (AFL) is introduced by Agrawal et al. [26]. The algorithm utilized a temporal weighted averaging approach, estimating the number of clients expected in each round and continuously updating the global model with a weighted average of received parameters. This approach avoided deadlock issues and increased server and client throughput. The algorithm is evaluated on an IDS dataset called NSL-KDD, outperforming traditional FL models in terms of accuracy and performance. The global model achieved approximately 99.5% accuracy on the dataset, surpassing traditional FL models in anomaly detection. Additionally, the algorithm showed an increased throughput of around 10.17% for every 30 timesteps in terms of asynchronicity.

To defend against poisoning attacks on IoT networks, Zhang et al. [27] developed a secure FL-based NIDS method that employs two levels of defense mechanisms: model-level and data-level. The former identifies and discards compromised models, while the latter filters out malicious traffic data. The authors evaluated their method on two datasets, UNSW-NB15 and CICIDS2018, and reported significant accuracy improvements compared to other methods: up to 48% and 36% respectively with the model-level defense, and an additional 13% with the data-level defense on CICIDS2018.

In the context of Industrial Internet of Things (IIoT) systems, Huong et al. [3] proposed a cyberattack anomaly detection system using a model architecture comprising a VAE-Encoder, an LSTM unit, and a VAE-Decoder. They employed the Kernel Quantile Estimator (KQE) to learn an optimal threshold for high anomalous discrimination accuracy. The proposed technique achieves a 97.9% F1-score when evaluated on time series data obtained from a gas pipeline factory SCADA system. Additionally, the approach demonstrates a 35% reduction in bandwidth compared to centralized learning architectures.

The authors in [28] incorporated attention mechanisms into the FL model to create an intrusion detection mechanism. They used a simple CNN architecture to accommodate the constrained computational resources of the clients. A weighted version of federated averaging is utilized, with weights calculated based on the normalized distances between the global and local models. The proposed model achieves 99.65% accuracy in the centralized version of the intrusion detection system and 99.12% accuracy and 88.97% F1-score in its federated version.

B. DEEP LEARNING PRUNING TECHNIQUES

To reduce latency, training time, and energy consumption in Deep Neural Networks (DNNs), various approximation techniques have been developed, one of which is software-based DNN pruning. Pruning involves selectively removing connections, filters, and/or channels from the network based on their importance. There are two main types of pruning: structured (coarse-grained) [29], [30], [31] and unstructured (fine-grained) [32], [33]. Structured pruning involves removing entire groups of connections, filters, or channels from the network, while unstructured pruning removes individual connections or elements. By applying pruning, the network becomes more compact and less computationally intensive, leading to faster inference and reduced energy consumption. However, it is essential in real-world applications to carefully balance the trade-off between model size reduction and drops in performance when applying pruning techniques.

Su et al. [34] investigated the extent to which unstructured pruning depends on the training dataset and the model architecture. They evaluated the data dependency by applying corrupted data in the pruning step and original data in the retraining step. They also evaluated the architecture dependency of the pruning technique by randomly permuting all the connections of neurons layer-wise and thus obliterating the structure obtained in the pruning step. The obtained results reveal that the selected unstructured pruning technique has no dependence on the training data and the architecture of the model. Moreover, the authors improved the performance of partially-trained tickets by proposing a new pruning technique called hybrid tickets that combines both random tickets and partially-trained tickets.

Tanaka et al. [35] introduced a pruning technique for neural networks without any data by iteratively preserving synaptic flow. The authors aimed to find a pruned model that can be obtained at the initialization phase without training or seeing the data. They introduced a conservation law that reveals why existing gradient-based pruning methods encounter layer collapse during the initialization step and how to prevent it. They also proposed a new pruning method, called Iterative Synaptic Flow Pruning (SynFlow), that maintains the total flow of synaptic weights through the network during the initialization step under a sparsity constraint. They tested SynFlow with various models, datasets, and sparsity levels and showed that it matched or exceeded existing state-of-the-art pruning methods at the initialization phase.

Cai et al. [36] presented a structured pruning method for CNNs at the initialization phase without using any data or training. The authors strived to discover sparse subnetworks that can achieve high performance and efficiency on various datasets and architectures. PreCrop is the proposed technique that directly reduces the model size at the channel level according to the layerwise compression ratio. PreCrop is regular and dense in storage and computation, unlike weight pruning, and does not compromise accuracy. Furthermore, since PreCrop prunes CNNs during the initialization phase, the computational and memory costs of CNNs are lowered for the training and inference on common hardware. They tested their method on several modern CNN architectures, such as ResNet [19], MobileNetV2 [37], and EfficientNet [38] with two well-known datasets CIFAR-10 [39] and ImageNet [40]. They demonstrated that their method achieved higher accuracy than existing pruning at the initialization phase using the same FLOPs and even improved the accuracy of some models with fewer parameters.

Fang et al. [15] proposed a novel method for structural pruning of deep neural networks, called DeepGraph, that can handle any network architecture and pruning granularity. The method is based on constructing a dependency graph that captures the relationships between network components, such as layers, channels, filters, or neurons. The dependency graph then guides the pruning process, ensuring the pruned network is valid and functional. The paper also introduces a new metric, called structural sparsity ratio (SSR), to measure the degree of structural pruning. The paper also demonstrated the applicability of DeepGraph to other tasks, such as network quantization and distillation.

Unlike previously discussed works, our network intrusion detection method employs a zero-shot pruning technique with FL. The pruning technique ensures data privacy since it is able to select the most important connections without accessing clients' data. Therefore, our method combines the knowledge of multiple local network intrusion detection learning models into a single global model, ensuring reductions in both training and inference times and resource optimization.

III. LIGHTWEIGHT-FED-NIDS FRAMEWORK

Lightweight-Fed-NIDS framework is composed of two main components: 1) FL architecture for NIDS, and 2) a Lightweight and distributed method. In this section, we describe the main elements of our framework. We should notice that the parameters notation provided in Table 1 holds in the remainder of the paper.

A. LIGHTWEIGHT-FED-NIDS ARCHITECTURE

Lightweight-Fed-NIDS is designed to preserve data privacy, reduce computing resources, and build high-detection performance. This system architecture is designed to accommodate heterogeneous hardware and software components, which may share a common structure but possess distinct characteristics like storage size, processing power, and memory capacity. In our Lightweight-Fed-NIDS framework, the adoption of FL is fundamental to our approach for several critical reasons. Firstly, FL enables us to leverage distributed computational resources across multiple network nodes, allowing for more robust and scalable intrusion detection. More importantly, FL addresses a crucial challenge in modern network security: the need for privacy-preserving collaborative learning. In many network environments, sharing raw network traffic data for centralized analysis poses significant privacy and security risks. By employing FL, our system allows multiple network entities to collaboratively train a global intrusion detection model without exchanging sensitive local data. This approach is particularly valuable in scenarios where networks contain proprietary or confidential information that cannot be shared directly. Additionally, FL in our framework enables continuous model updating with minimal communication overhead, as only model parameters are shared instead of raw data. This is especially beneficial in dynamic network environments where threat landscapes evolve rapidly. Furthermore, the decentralized nature of FL in our system enhances its resilience against single points of failure and potential attacks on the central server. While traditional centralized approaches have their merits, we argue that the privacy-preserving, scalable, and adaptable nature of FL in Lightweight-Fed-NIDS offers unique advantages that are increasingly crucial in modern, distributed network security paradigms. Our system comprises two main parts: *NIDS clients* and *NIDS global server* as illustrated in Figure 1.

TABLE 1. FL architecture parameters notation and description.

Notation	Description
θ_0	The initial model's parameters.
θ	The global model's parameters.
θ_j	The local model for the NIDS client j.
θ_{t+1}	The global model's parameters for round $t+1$.
c_j	NIDS client j.
C	The set of the NIDS participants.
N	The number of participating NIDS clients in a single round.
T	The total number of communication rounds.
$\mathcal{M}$	The pruning mask.
N_j	The size of the dataset of the NIDS client j.
D_j	The dataset of the NIDS client j.
B	The batch size.
η	The learning rate.
$\mathcal{L}$	The loss function.

1) NIDS CLIENTS

We consider that each client j is a NIDS that performs the following pipeline:

1) Collecting local data D_j from local network traffic via sensors or devices (PLCs, RTUs, or SCADA systems),
2) Applying pruning techniques using the received mask $\mathcal{M}$ and model weights θ,
3) Optimizing local model θ_j with local data D_j for intrusion detection, and
4) Sending updated models to the *NIDS global server*.

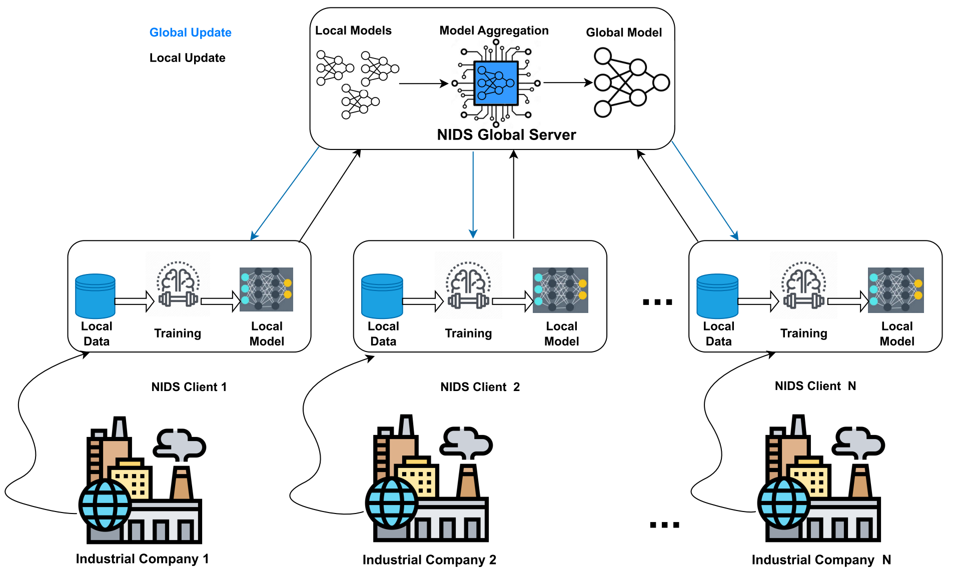

FIGURE 1. **Lightweight-Fed-NIDS architecture.**

2) NIDS GLOBAL SERVER

This component plays two primary roles: system initialization and model aggregation.

1) **System initialization.** The initialization step involves setting the model weights θ_0 and the pruning mask $\mathcal{M}$ and then disseminating them to the NIDS clients. It is essential to emphasize that we strategically chose to dispatch both the model and the mask to each client for the following reasons: clients are unable to effectively optimize the pruned model, as each training phase necessitates access to both the model and the pruning mask. This requirement emerges from the structural modifications that the model undergoes during the pruning process. Therefore, the server undertakes the responsibility of determining the pruning mask, thereby ensuring that essential computations are managed on the server side. It is pertinent to mention that we dispatch the mask only once.
2) **Model Aggregation.** In the aggregation step, the received updated and pruned clients' models $\{\theta_1, \theta_2, \ldots, \theta_N\}$ are aggregated using FedAVG[1] [7]. In general, the process of receiving and aggregating clients' models is repeated until a predefined criterion is met. In our case, we define the number of communication rounds T as the stop criteria.

B. LIGHTWEIGHT-FED-NIDS METHODS

In the following, we describe the main steps we apply to build Lightweight and distributed ML models for NIDS. Figure 2 outlines the applied workflow that connects these steps, including initialization, local model optimization, and model aggregation.

[1]Averaging all the clients' model weights.

1) INITIALIZATION

The initialization step consists of two phases: model definition and pruning mask computation.

a: MODEL DEFINITION

To start with, assuming that a secure channel is established between the *NIDS global server* and the *NIDS clients*, we define the model architecture and initialize its parameters. To address the vanishing and exploding gradient problems, we apply the Kaiming He technique to initialize the model's weights [41]. The architecture is composed of two blocks, including a feature extractor and a classifier. The feature extractor applies multiple convolution layers to extract relevant features from network flows. A convolution layer is composed of four consecutive steps: 1) convolution operation; 2) batch normalization; 3) ReLU activation function; and 4) pooling. Given that we represent a network flow as a 3-channel image (more details can be found in subsection III-B2), the convolution operation is defined by the following equation:

$$y_{i^{l+1},j^{l+1},d} = \sum_{i=0}^{H-1} \sum_{j=0}^{W-1} \sum_{d^l=0}^{\mathcal{K}^l-1} \theta_{i,j,d^l,d} \times x^l_{i^l+i,j^l+j,d^l} \tag{1}$$

where:

- $y_{i^{l+1},j^{l+1},d}$ represent a single spatial location's computation within the convolutional layer $l+1$ at position (i^{l+1}, j^{l+1}) and depth d in the $(l+1)$-th layer.
- $x^l_{i^l+i,j^l+j,d^l}$ represents the input value at position $(i^l + i, j^l + j)$ and depth d^l in the l-th layer.

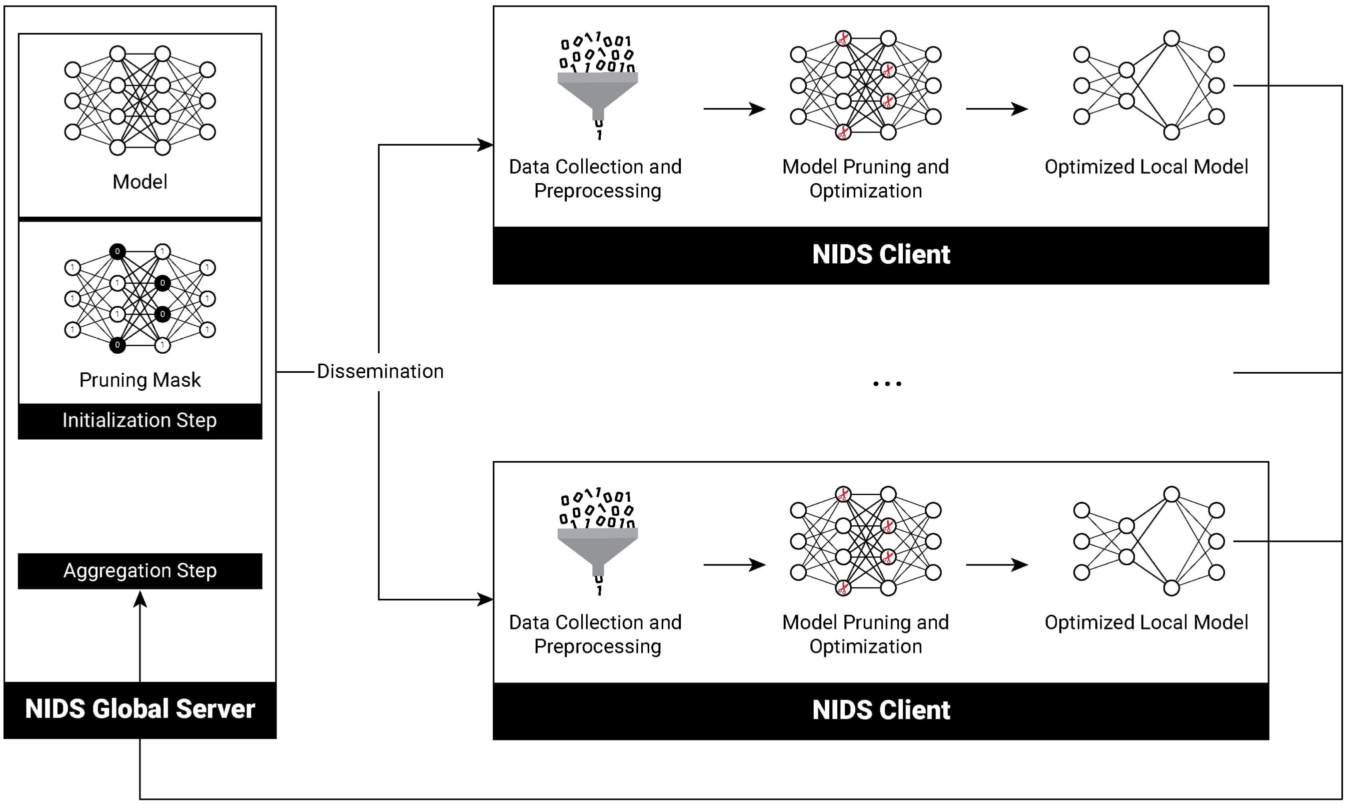

FIGURE 2. The flowchart of Lightweight-Fed-NIDS methods.

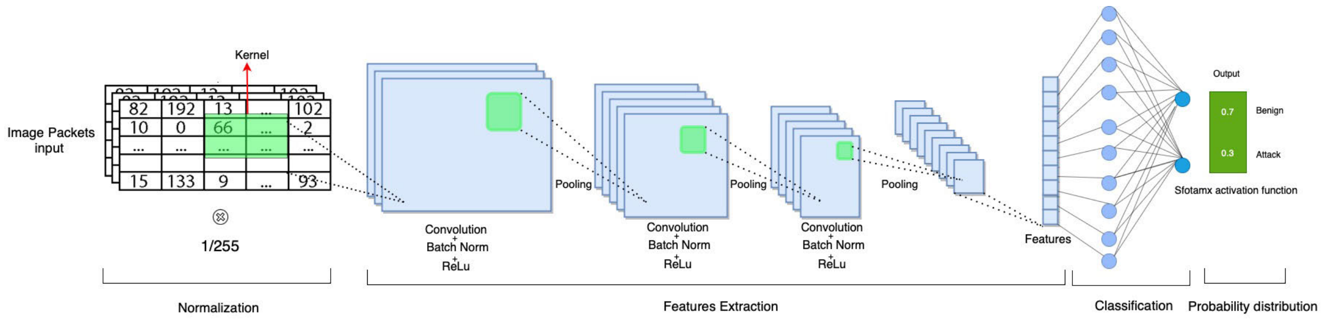

FIGURE 3. The main blocks of the model architecture of the NIDS Model.

- $\theta_{i,j,d^l,d}$ represents the convolutional kernel (or filter) parameter connecting the input depth d^l in the l-th layer to depth d in the $(l+1)$-th layer.
- H and W are the height and width of the convolutional kernel, respectively.
- $\mathcal{K}^l$ represents the number of input depth channels in the l-th layer.

Batch Normalization [42] addresses the internal covariate shift challenge and is defined as follows:

$$\mathrm{BN}(X) = \gamma \odot \frac{X - \mu_B}{\sigma_B} + \beta \tag{2}$$

where X is the input feature map to the convolutional layer. μ_B and σ_B are the mean and standard deviation of X computed along the batch dimension and the spatial dimensions. γ and β are learnable parameters that scale and shift the normalized output. $\odot$ denotes element-wise multiplication.

The ReLU activation function is described by equation (3):

$$ReLU(x) = max(0, x) \tag{3}$$

The Max-Pooling function is described as follows:

$$\mathrm{MaxPooling}(X)_{i,j,k} = \max_{m,n} X_{i\cdot s_x+m, j\cdot s_y+n,k} \tag{4}$$

where X is the input, (i, j) are the indices of the output, k is the channel index, s_x and s_y are the stride values in the horizontal and vertical directions, respectively, and the pooling window is defined by the filter size f_x and f_y centered at the output index (i, j).

We use three well-known computer-vision models, namely, ResNet-50 [19], ResNet-101 [19], and VGG-19 [20] as feature extractors, allowing richer feature extraction from network flow images. The feature vector is then fed into a fully connected layer, followed by a softmax activation function (Equation (5)) to produce the final class probabilities. Figure 3 depicts the full architecture of our neural network model. Next, the *NIDS global server* defines the loss function $\mathcal{L}$ (Equation (6)), and initializes the model's parameters $\theta_0 = \{\theta_0^{l_1}, \theta_0^{l_2}, \cdots, \theta_0^{l_{L+1}}\}$ for the deep learning intrusion detection model, the learning rate η, the batch size B, and the total number of communication rounds T.

$$p_i = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}} \tag{5}$$

$$\mathcal{L} = -\sum_{i=1}^{N} y_i \log(p_i) + (1 - y_i)\log(1 - p_i) \tag{6}$$

where N is the number of samples, y_i is the ground truth of the i-th sample, and p_i is the predicted probability for the i-th sample.

b: PRUNING MASK COMPUTATION

Once the model's weights are initialized, we employ the zero-shot and structured pruning technique to calculate the pruning mask. This mask is applied during the initialization step and does not require training data, as demonstrated by Cai et al. [36]. This approach eliminates the need to create the mask through multiple iterations, saving time and resources. We define the mask as a tensor $\mathcal{M} \in \mathbb{T}^{l_1 \times l_2 \times \cdots \times l_{L+1}}$ such that the values of $\mathbb{T} \in \{0, 1\}$. Similarly to θ, we represent $\mathcal{M}$ as a set of tensors $\{\mathcal{M}^{l_1}, \mathcal{M}^{l_2}, \cdots, \mathcal{M}^{l_{L+1}}\}$. The intuitive idea is that zero values of $\mathcal{M}_{ji}^{l}$ highlight the parameters of θ that should be pruned, while a one value means that the weight is retained. Fang et al. [15] claimed that it is inefficient to prune some parameters of a certain layer of a CNN model without considering its dependencies on other layers in the network. Therefore, we applied the inter-layer and intra-layer dependencies criteria introduced in [15] to group layers. After that, a score of each group is computed to identify and prune unimportant parameters. The score is defined as follows:

$$\hat{I}_{g,k} = N \cdot I_{g,k} / \sum \{\mathrm{TopN}\left(I_g\right)\} \tag{7}$$

where:

- $g = \{\theta_1, \theta_2, \ldots, \theta_{|g|}\}$ represents the parameter group.
- $I(\theta) = \|\theta\|_1$ represents the L_1-norm.
- $I_{g,k} = \sum_{\theta \in g} \|\theta[k]\|_1^2$ represents the importance of the k-th prunable dimensions.
- $I(g) = \sum_{\theta \in g} I(\theta)$: represents the group importance score.

To compute the importance of each group of parameters, we rely on the L_1-norm, which measures the magnitude of each parameter group. The importance score of each prunable dimension is calculated and normalized to guide the pruning process. Groups with lower importance scores are pruned according to a predefined pruning rate, ensuring that the least important parameters are eliminated while preserving those that are crucial for model accuracy.

Then, using a predefined pruning rate, the mask is generated, leading to a smaller and lighter model. The resulting pruning mask is a binary tensor of the same size as the model. The model is pruned by applying element-wise matrix multiplication between the pruning mask and the model's weights. The next equation defines the new pruned model's weights θ':

$$\theta' = \mathcal{M} \odot \theta_0 \tag{8}$$

It is worth mentioning that the zero-shot method offers minimal practical advantages since it is commonly restricted to simply setting weights to zero without eliminating the pruned parameters. This means that the model size and computational cost remain unchanged, and the benefits of sparsity are not fully realized. Consequently, we have incorporated the DeepGraph technique[2]to accelerate sparse tensor multiplications and free up memory by removing zeroed parameters. Moreover, the pruning technique is model-dependent and deterministic, indicating that the outcome of pruning is inherently tailored to the specific architecture of the model in question. Finally, the NIDS server disseminates the initialized model, training parameters, and the pruning mask to the participating NIDS clients.

2) MODEL OPTIMIZATION

NIDS clients are tasked with optimizing the pruned model using their local data. Each *NIDS client* collects raw network packets through sensors and processes them using NFStream [43]. This tool constructs network flows by grouping raw packets based on the 5 tuple fields: IP source, IP destination, source port, destination port, and protocol. Next, an NIDS client applies an internally developed NFStream plug-in to convert the collected flows into matrices based on raw packet bytes. It considers only the first 300 bytes from each packet and limits the selection to the first 40 packets [44]. The 300 bytes correspond to 20% of the packet at the IP level (MTU=1500 in Ethernet Networks). This approach differs from existing literature that typically utilizes smaller data sizes; for example, Sun et al. [45] used 40 bytes, while both Liu et al. [46] and Pham et al. [47] employed a 30 bytes strategy. Our choice of a larger input size is aimed at assessing the efficiency of pruning techniques with more extensive data sizes and capturing the most significant information from larger packets. This input size can be further tuned (beyond the scope of the paper). To standardize the lengths of flows, padding or truncation is applied as necessary. Following this, the NIDS client builds 3-channel images by replicating network flow matrices three times. Finally, it normalizes the pixel values of the generated images to a common range, enhancing comparability and reducing bias in subsequent processing

[2]https://github.com/VainF/Torch-Pruning

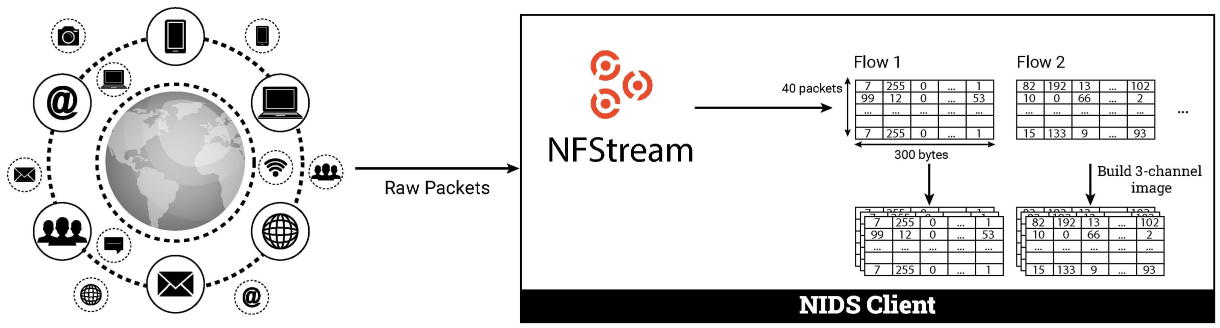

FIGURE 4. Raw packets preprocessing step performed by each NIDS client.

and modeling. Figure 4 illustrates the data preprocessing step carried out by NIDS clients.

In the next stage, the model is optimized using local network traffic data to distinguish between normal flows and intrusions. This optimization process involves refining the local model over several epochs to achieve optimal accuracy and performance. Algorithm 5 outlines the steps performed by the NIDS client to optimize its local model, which includes incorporating information from the global model, applying the pruning mask computed at the initialization step, and conducting gradient descent-based training. Lastly, the NIDS client transmits the updated and pruned model to the NIDS global server to build the Lightweight NIDS model.

```
Require: Global model $\theta$ and the mask $\mathcal{M}$
Ensure: Local model updated $\theta$
 1: $B \leftarrow$ The set of batches
 2: $E \leftarrow$ The number of epochs
 3: $\eta \leftarrow$ The learning rate
 4: Receive the global model $\theta$ and pruning mask $\mathcal{M}$ from
    the server
 5: Apply the mask to the model to get the pruned model:
    $\theta \leftarrow \theta \odot \mathcal{M}$
 6: for epoch from 1 to E do
 7:     for each $b \in B$ do
 8:         $\theta \leftarrow \theta - \eta \nabla \mathcal{L}(\theta, b)$
 9:     end for
10: end for
```

FIGURE 5. NIDS client model optimization.

3) MODEL AGGREGATION

During each communication round, the server aggregates the parameters received from the NIDS clients using the Federated Averaging (FedAVG) algorithm [7]. FedAVG works by averaging the local model updates from the participating clients. More formally, let θ_i^t be the model parameters of client i at round t. The global model update θ_{global}^{t+1} at round $t+1$ is calculated as follows:

$$\theta_{global}^{t+1} = \frac{1}{N} \sum_{i=1}^{N} \theta_i^t \tag{9}$$

This iterative process continues for a predefined number of rounds T. After each round, the most recent updated global model is distributed to all NIDS clients within the network. These clients incorporate and replace their local models with the received global model, ensuring that each client benefits from the collective knowledge of the entire network. This procedure enhances the intrusion detection system's capability by enabling the distributed clients to collaboratively learn patterns from diverse data sources, leading to a more generalized and robust anomaly detection model. Furthermore, as the clients continually synchronize their local models with the global model, the system becomes better equipped to detect new and emerging threats.

```
Require: Clients $C = \{c_1, c_2, ..., c_N\}$
Ensure: Global model $\theta$
 1: $N \leftarrow$ The number of participating clients
 2: $T \leftarrow$ The Number of rounds
 3: The Server initializes the global model $\theta_0$ and the mask
    $\mathcal{M}$.
 4: The Server broadcasts the global model $\theta_0$ and the mask
    $\mathcal{M}$ to all clients.
 5: for round $t$ from 1 to $T$ do
 6:     for $j$ from 1 to $N$ do
 7:         Receive $\theta_{t,j}$ from Client $c_j$
 8:     end for
 9:     Compute $\theta_{t+1} \leftarrow \frac{1}{N} \sum_{j=1}^{N} \theta_{t,j}$    ▷ Server aggregates
    the received models from all clients.
10:     Server broadcasts the updated model $\theta_{t+1}$ to all
    clients.
11: end for
```

FIGURE 6. NIDS global server pipeline.

Finally, the aggregated global model is distributed to the clients, who use it to replace their local models and continue the training process. Algorithm 6 presents a thorough description of the pipeline performed by the NIDS global server.

IV. EXPERIMENTAL RESULTS

This section provides all the necessary information regarding the conducted experiments, encompassing datasets, experimental settings, and performance evaluation. These experiments aim to address three primary research questions, namely:

1) **RQ1:** Can pruning be easily or conveniently applied to centralized NIDS?
2) **RQ2:** Is it feasible to optimize a non-optimized and zero-shot pruned NIDS model using FL in a distributed environment?
3) **RQ3:** What is the impact of the pruning technique on training and inference complexities?

A. DATASETS

We employ three distinct datasets to assess the Lightweight-Fed-NIDS framework. These datasets are well-suited for external communication networks and are widely utilized within the NIDS community. We provide a description of each dataset below.

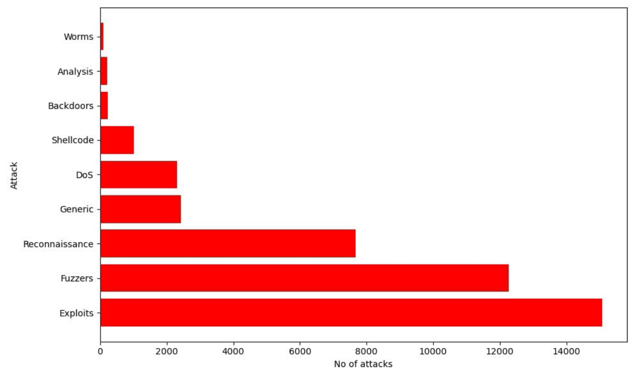

FIGURE 7. Attacks type distribution in UNSW-NB15.

- **UNSW-NB15** [22] was collected in 2015 by the School of Engineering and IT at UNSW Canberra at ADFA. The authors used a small network simulation for 31 hours to collect normal and malicious network packets. The dataset has nine types of attacks: analysis, backdoors, DoS, exploits, generic, fuzzers, reconnaissance, shell code, and worms. It has more than two million records with 49 features of different types and comes also in pcap format. The data distribution of the dataset is shown in Figure 7.
- **USTC-TFC2016** [23], released in 2017, contains ten kinds of malware traffic from public websites that were captured from a real network environment between 2011 and 2015. The dataset also includes ten kinds of normal traffic generated using IXIA BPS, a professional network traffic simulation tool. The dataset is 3.71 GB in size and is in the pcap format. Figure 8 shows the dataset's distribution.
- **CIC-IDS-2017** dataset was created by the Canadian Institute for Cybersecurity (CIC) [21] in a simulated network setting. It consists of 5 days (July 3 to July 7, 2017) of network traffic with various common attack types such as FTP patator, SSH patator, DoS slowloris, DoS Slowhttptest, DoS Hulk, DoS GoldenEye, Heartbleed, Brute force, XSS, SQL Injection, Infiltration, Bot, DDoS and Port Scan. Each attack type has 80 features extracted by CICFlowMeter [24], [25]. The dataset also contains full packet payloads in pcap format. Figure 9 shows the dataset's distribution.

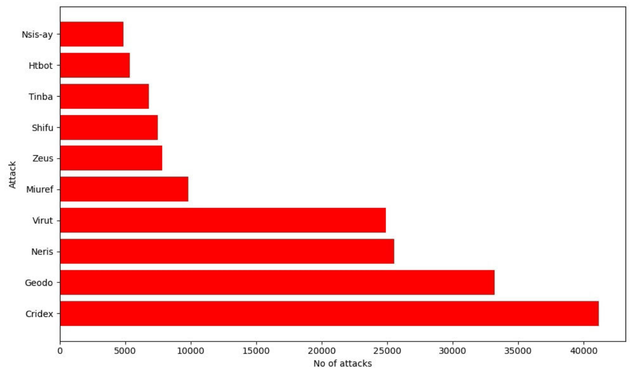

FIGURE 8. Attacks type distribution in USTC-TFC2016.

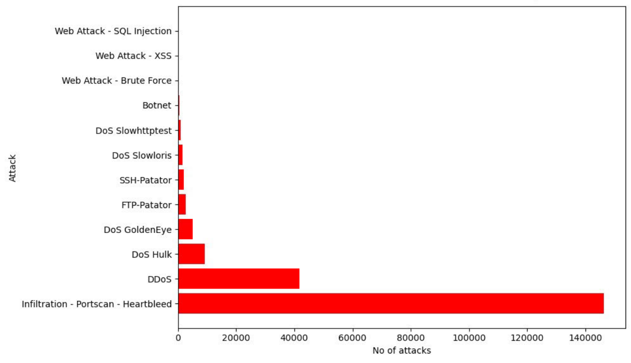

FIGURE 9. Attacks type distribution in CIC-IDS-2017.

We evenly distributed the datasets previously described among all participants during the experiments using the Independent and Identically Distributed (IID) approach. This approach divided normal and attack samples proportionally among all participating NIDS clients. The goal was to ensure that each participant received an equal number of samples from each attack type as well as normal data. As a result, we achieved a balanced and equitable distribution of data across all participants, facilitating a representative and accurate analysis. Figure 10 depicts each dataset's distribution of benign and attack samples.

B. EXPERIMENTAL SETTINGS

We implemented the Lightweight-Fed-NIDS framework in Python 3.9.15 using the following libraries: NFStream, Numpy, Pandas, PyTorch, and Torch-Pruning.[3] We allocated 70% of the data for training and 30% for testing, with the

[3]https://github.com/VainF/Torch-Pruning

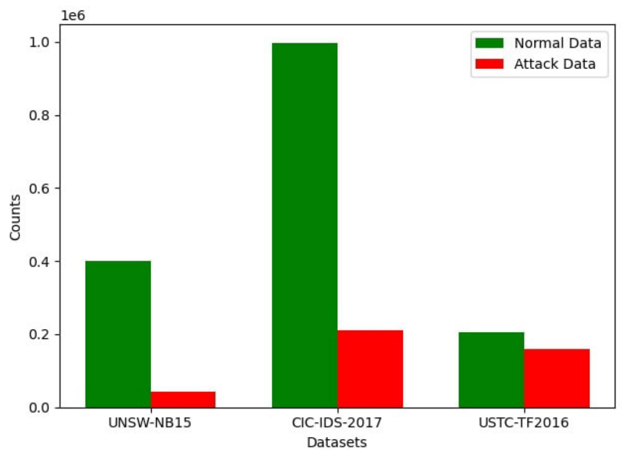

FIGURE 10. Normal and attack data distributions in UNSW-NB15, CIC-IDS-2017, and USTC-TF2016 datasets.

testing set reserved for final evaluation. The training set was further randomly divided into an 80% training set and a 20% validation set. The hyperparameters employed for training the local NIDS model consisted of ten epochs and a learning rate of 2×10^{-4}.

In the federated learning phase, we set the communication rounds value T to 5, as the model ceased to evolve beyond this round. We then assessed the performance of our method with 10, 50, and 100 participants. Notably, at the aggregation step, we only select 50% of the participating NIDS clients to perform the aggregation. In the model pruning process, we have tried different sparsity levels, including 50%, 70%, and 90%. We use convenient metrics to evaluate the performance of our methods, including:

- Training Accelerating (TA),
- Inference Acceleration (IA),
- ΔAcc: the accuracy difference between pruned and unpruned model,
- Δ F1-sc: F1-score difference between pruned and unpruned model,
- F1-score and accuracy.

It is important to note that our study's training and inference time measurement is conducted at the client level. Specifically, we calculate these times at the end of each communication round by measuring how long each client takes to complete its portion of the training and inference processes. To provide a comprehensive view of the system's performance, we then calculate the average of these times across all clients participating in the FL process. Our methodology includes all communication rounds in the FL cycle. This means that the training and inference times reported are not limited to a single communication period (T), but rather encompass the entire series of interactions between the server and clients throughout the learning process.

Finally, our experiments were conducted on a single node within the African Super-Computing Center HPC cluster equipped with four A100 SXM4 80 GB GPUs from NVIDIA and supported by Mohammed VI Polytechnic University.

C. PERFORMANCE EVALUATION

1) RESEARCH QUESTION 1

To address the first research question **RQ1**, we simulated a scenario in which all data were centralized. We conducted a series of experiments to compare the unpruned NIDS models (baseline) with the pruned NIDS models. Tables 2, 3, and 4 present the results of the centralized settings for each dataset, respectively. Our findings indicate that pruning does not compromise the performance of the models; on the contrary, some pruned models achieve slightly better results than their non-pruned counterparts. For instance, in Table 2, we can observe improvements in accuracy and F1-score compared to the baseline (unpruned model). When using ResNet-50 as a feature extractor with 70% pruning rate, the model achieved 99% accuracy and F1-score, representing a 0.01% improvement in performance. In most cases, we found that the model did incur degradation, meaning that it either maintained or improved its performance. However, in a few cases, we observed a slight degradation of 10^{-4} in accuracy. Therefore, pruning benefits the model by maintaining or increasing detection performance with no added overhead. These same conclusions apply to Tables 3 and 4, indicating that pruning is efficiently applied to centralized NIDS.

2) RESEARCH QUESTION 2

The next research question **RQ2** is related to the impact of the pruning technique on NIDS performance in federated learning settings. The results are presented in Tables 5, 6, and 7. Our Lightweight-Fed-NIDS framework with a pruning technique demonstrates the ability to achieve high accuracy and F1-score on all three datasets, especially on the USTC-TFC2016 dataset, where the accuracy and F1-score are above 0.99% in most cases. The results also show that the pruning technique can significantly reduce the training and inference time. For instance, using VGG-19 as a feature extractor, we achieve up to 3.5X speedup in training time and 2.5X speedup in inference time across all datasets while maintaining comparable detection performance in most cases. The optimal sparsity level depends on the feature extractor and the dataset. For instance, in the case of the CIC-IDS-2017 dataset both ResNet-50 and ResNet-101 (Table 5) achieve the best performance with 70% sparsity, exhibiting an increase in model accuracy by 3×10^{-4}. In contrast, VGG-19 achieves the best performance with 50% sparsity in accuracy and F1-score. It is worth mentioning that different feature extractors have different characteristics and suitability for different datasets. For example, on the UNSW-NB15 (Table 6) dataset, VGG-19 outperforms ResNet-50 and ResNet-101 in terms of accuracy and F1-score, while on the USTC-TFC2016 dataset (Table 7), ResNet-50 and ResNet-101 outperform VGG-19 in terms of accuracy and F1-score. Therefore, choosing an appropriate feature

TABLE 2. Centralised result for CIC-IDS-2017.

Feature extractor	Sparsity	accuracy	F1-score	Training Time	Inference Time	TA	IA	Δ Acc	Δ F1-sc
ResNet-50	0	0.9993	0.9993	2147.88	495.74				
	50	0.9994	**0.9995**	1149.37	365.97	× 1.87	× 1.35	+ 0.0001	+ 0.0002
	70	0.9994	0.9994	1128.95	337.91	× 1.9	× 1.47	+ 0.0001	+ 0.0001
	90	0.9989	0.9987	694.11	272.2	× 3.09	× 1.82	− 0.0004	− 0.0006
ResNet-101	0	0.9985	0.9989	3393.54	733.1				
	50	0.9996	0.9993	1801.36	524.18	× 1.88	× 1.4	+ 0.0004	+ 0.0004
	70	0.9991	0.9992	1710.12	476.83	× 1.98	× 1.54	+ 0.0006	+ 0.0003
	90	0.999	0.9983	1085.96	386.65	× **3.12**	× **1.9**	+ 0.0005	− 0.0006
VGG-19	0	0.9993	0.9994	1104.04	369.46				
	50	**0.9997**	0.9994	580.65	276.76	× 1.9	× 1.33	+ 0.0004	0
	70	0.9994	0.9994	571.9	267.78	× 1.93	× 1.38	+ 0.0001	0
	90	0.9989	0.9988	440.41	199.1	× 2.51	× 1.86	− 0.0004	− 0.0006

TABLE 3. Centralised result for UNSW-NB15.

Feature extractor	Sparsity	Accuracy	F1-score	Training Time	Inference Time	TA	IA	Δ Acc	Δ F1-sc
ResNet-50	0	0.9731	0.9533	402.55	104.12				
	50	0.973	**0.9536**	195.83	64.28	× 2.06	× 1.62	+ 0.0001	− 0.0003
	70	0.9557	0.9499	206.44	65.24	× 1.95	× 1.6	+ 0.0174	+ 0.0034
	90	0.9712	0.9461	95.85	38.89	× **4.2**	× **2.68**	+ 0.0019	+ 0.0072
ResNet-101	0	0.9591	0.9496	736.31	151.21				
	50	0.9775	0.9535	372.07	105.37	× 1.98	× 1.44	− 0.0039	− 0.0039
	70	**0.9779**	0.9503	363.32	91.6	× 2.03	× 1.65	− 0.0188	− 0.0007
	90	0.9692	0.9437	213.25	67.92	× 3.45	× 2.23	− 0.0101	+ 0.0059
VGG-19	0	0.9649	0.9494	1212.22	240.17				
	50	0.9783	0.953	600.59	162.97	× 2.02	× 1.47	− 0.0134	− 0.0036
	70	0.9788	0.9479	580.4	141.81	× 2.09	× 1.69	− 0.0139	+ 0.0015
	90	0.9765	0.9459	351.14	112.53	× 3.45	× 2.13	− 0.0116	+ 0.0035

TABLE 4. Centralised result for USTC-TFC2016.

Feature Extractor	Sparsity	Accuracy	F1-score	Training Time	Inference Time	TA	IA	Δ Acc	Δ F1-sc
ResNet-50	0	0.9999	0.9999	612.62	125.77				
	50	1	1	306.93	86.36	× 2	× 1.46	− 0.0001	− 0.0001
	70	**1**	**1**	297.86	75.44	× 2.06	× 1.67	− 0.0001	− 0.0001
	90	0.9998	0.9998	213.49	82.17	× 2.87	× 1.53	+ 0.0001	+ 0.0001
ResNet-101	0	0.9997	0.9996	1004.92	198.54				
	50	0.9996	0.9996	508.32	136.35	× 1.98	× 1.46	+ 0.0001	0
	70	0.6932	0.6838	485.3	118.76	× 2.07	× 1.67	+ 0.3065	+ 0.3158
	90	0.994	0.9946	285.82	92.29	× **3.52**	× 2.15	+ 0.0057	+ 0.005
VGG-19	0	1	1	341.75	108.61				
	50	1	1	168.54	65.34	× 2.03	× 1.66	0	0
	70	0.9996	0.9996	178.42	67.54	× 1.92	× 1.61	+ 0.0004	+ 0.0004
	90	0.9965	0.9963	102.85	45.01	× 3.32	× **2.41**	+ 0.0035	+ 0.0037

extractor is important for achieving good performance in federated learning settings.

To further investigate the impact of the pruning technique on the FL performance, we conducted a comparative analysis of the convergence of the model based on the VGG-19 feature extractor with and without pruning on three datasets using multiple sparsity levels. Figures 11, 12, and 13 depict the convergence of the VGG-19 feature extractor model with all datasets, including CIC-IDS-2017, UNSW-NB15, and USTC-TFC2016. These graphs show how the sparsity level influences the model's behavior during the training rounds. We can see that increasing the sparsity level causes some fluctuations in the model's performance in the first rounds. However, as training progresses, the model adapts and becomes more stable, eventually converging in the final round. For a detailed evaluation of the convergence of models using ResNet-50 and ResNet-100 feature extractors, please refer to **Appendix**.

Our proposed approach achieves performance that closely aligns with the centralized approach across different datasets. This underscores the trustworthiness and practicality of the Lightweight-Fed-NIDS framework. These findings highlight

TABLE 5. Performance of our FL approach on the CIC-IDS-2017 dataset with 10 users.

Feature Extractor	Sparsity	Accuracy	F1-score	Training Time (s)	Inference Time (s)	TA	IA	Δ Acc	Δ F1-sc
ResNet-50	0	0.9991	0.9985	216.7915	23.4688				
	50	0.9992	0.9987	115.4494	17.2402	× 1.88	× 1.36	+ 0.0001	+ 0.0002
	70	0.9994	0.9992	112.0796	15.5866	× 1.93	× 1.51	+ 0.0003	+ 0.0007
	90	0.9982	0.9976	68.9205	13.0768	× 3.15	× 1.79	− 0.0009	− 0.0009
ResNet-101	0	0.9982	0.9976	335.0506	32.7349				
	50	0.9993	0.9991	177.5997	24.6819	× 1.89	× 1.33	+ 0.0015	+ 0.0015
	70	0.9994	**0.9993**	165.1246	20.894	× 2.03	× 1.57	+ 0.0012	+ 0.0017
	90	0.9976	0.9971	96.1629	15.3922	× 3.48	× 2.13	− 0.0006	− 0.0005
VGG-19	0	0.9994	0.9989	111.1094	17.1264				
	50	**0.9995**	0.9993	56.0593	12.4499	× 1.98	× 1.38	+ 0.0001	+ 0.0004
	70	0.9993	0.9991	56.7763	11.491	× 1.96	× 1.49	− 0.0001	+ 0.0002
	90	0.9973	0.995	30.6895	6.0139	× **3.62**	× **2.85**	− 0.0021	− 0.0039

TABLE 6. Performance of our FL approach on the UNSW-NB15 dataset with 10 users.

Feature Extractor	Sparsity	Accuracy	F1-score	Training Time (s)	Inference Time (s)	TA	IA	Δ Acc	Δ F1-sc
ResNet-50	0	**0.9864**	0.947	85.9364	8.5626				
	50	0.946	0.944	49.9638	5.9962	× 1.72	× 1.43	− 0.0404	− 0.003
	70	0.956	0.947	49.0995	5.6498	× 1.75	× 1.52	− 0.0304	0
	90	0.967	0.9469	27.4747	2.5401	× 3.13	× **3.37**	− 0.0194	− 0.0001
ResNet-101	0	0.9546	0.9462	125.3833	12.8802				
	50	0.9602	0.9475	76.0897	9.1991	× 1.65	× 1.4	+ 0.0013	+ 0.0013
	70	0.9578	0.9456	75.917	9.2613	× 1.65	× 1.39	+ 0.0032	− 0.0006
	90	0.9657	0.9467	51.0107	7.0727	× 2.46	× 1.82	+ 0.0111	+ 0.0005
VGG-19	0	0.9451	0.9474	41.2214	5.683				
	50	0.9498	**0.9483**	20.9921	3.8341	× 1.96	× 1.48	+ 0.0047	+ 0.0009
	70	0.9526	0.9476	22.2571	4.1218	× 1.85	× 1.38	+ 0.0075	+ 0.0002
	90	0.9707	0.9446	12.1112	2.4126	× **3.4**	× 2.36	+ 0.0256	− 0.0028

TABLE 7. Performance of our FL approach on the USTC-TFC2016 dataset with 10 users.

Feature Extractor	Sparsity	Accuracy	F1-score	Training Time (s)	Inference Timev(S)	TA	IA	Δ Acc	Δ F1-sc
ResNet-50	0	0.9558	0.9605	68.5705	6.4813				
	50	0.9997	0.9997	40.4729	4.4137	× 1.69	× 1.47	+ 0.0439	+ 0.0392
	70	0.9997	0.9997	39.1443	4.1828	× 1.75	× 1.55	+ 0.0439	+ 0.0392
	90	0.9996	0.9996	21.0202	1.7252	× 3.26	× 3.76	+ 0.0438	+ 0.0391
ResNet-101	0	0.9997	0.9997	117.8598	10.6631				
	50	0.9995	0.9995	60.5672	6.6555	× 1.95	× 1.6	− 0.0002	− 0.0002
	70	0.9996	0.9997	60.3139	6.579	× 1.95	× 1.62	− 0.0001	0
	90	0.9991	0.9992	40.8388	5.2115	× 2.89	× 2.05	− 0.0006	− 0.0005
VGG-19	0	**0.9998**	**0.9998**	33.1345	4.4547				
	50	0.9995	0.9996	16.7902	2.9346	× 1.97	× 1.52	− 0.0003	− 0.0002
	70	0.9994	0.9995	17.8359	3.2198	× 1.86	× 1.38	− 0.0004	− 0.0003
	90	0.9962	0.9965	9.3135	1.7447	× **3.56**	× 2.55	− 0.0036	− 0.0033

the importance and feasibility of our framework in real-world scenarios, especially in distributed environments where multiple factories are managed under a unified organizational structure.

3) RESEARCH QUESTION 3

The last research question **RQ3** sheds light on the impact of the pruning technique on the training and inference complexities in both centralized and distributed settings. Our proposal is tailored to run on NIDS-client with limited resources. Consequently, we simulate two more complex scenarios where the numbers of NIDS clients are 50 and 100. We measured the training and inference times to see how pruning affected the computing performance. Table 8 presents the obtained results with all datasets when the model incorporates the ResNet-50 as a feature extractor. It is obvious that increasing the sparsity level for all feature extractors reduces the training time and inference time, meaning that pruning speeds up the model's training and inference complexities. However, the model's performance drops in the case of UNSW-NB15 with a 90% sparsity level. This can be due to the fact of the small and imbalanced

TABLE 8. Performance of Our FL Approach using ResNet-50 model with 50 Users.

Dataset	Sparsity	Accuracy	F1-score	Training Time (s)	Inference Time (s)	TA	IA	Δ Acc	Δ F1-sc
UNSW-NB15	0	0.966	0.9427	15.9736	1.8568				
	50	0.9664	0.9443	8.4761	1.4129	× 1.88	× 1.31	+ 0.0004	+ 0.0016
	70	0.9813	0.9387	8.1669	1.306	× 1.96	× 1.42	+ 0.0153	− 0.004
	90	0.5	0.474	4.8027	1.0287	× 3.33	× 1.8	− 0.466	− 0.4687
USTC-TFC2016	0	0.9964	0.9967	12.8327	1.3319				
	50	0.9962	0.9964	6.3505	0.8896	× 2.02	× 1.5	− 0.0003	− 0.0003
	70	0.982	0.9806	6.3083	0.8873	× 2.03	× 1.5	− 0.0144	− 0.0161
	90	0.933	0.9344	3.5337	0.5707	× 3.63	× **2.33**	− 0.0634	− 0.0623
CIC-IDS-2017	0	0.9974	0.9969	42.7476	4.5878				
	50	**0.9977**	**0.9974**	21.4627	3.0425	× 1.99	× 1.51	+ 0.0003	+ 0.0005
	70	0.9961	0.9962	23.3673	3.3751	× 1.83	× 1.36	− 0.0013	− 0.0007
	90	0.9869	0.981	15.3541	2.898	× 2.78	× 1.58	− 0.0105	− 0.0159

TABLE 9. Performance of our approach on CIC-IDS-2017 dataset using VGG-19 model and 100 users.

Sparsity	Accuracy	F1-score	Training Time (s)	Inference Time (s)	TA	IA	Δ Acc	Δ F1-sc
0	0.9952	0.9931	21.201	2.2541				
50	**0.9958**	**0.9938**	11.0328	1.6421	× 1.92	× 1.37	+ 0.0006	+ 0.0007
70	0.9898	0.9888	11.5533	1.651	× 1.84	× 1.37	− 0.0054	− 0.0043
90	0.9711	0.9649	7.4548	1.4296	× **2.84**	× **1.58**	− 0.0241	− 0.0282

TABLE 10. Performance of our approach on USTC-TFC2016 dataset using ResNet-50 model and 100 users.

Sparsity	Accuracy	F1-score	Training Time (s)	Inference Time (s)	TA	IA	Δ Acc	Δ F1-sc
0	0.9652	0.9628	6.585	0.8216				
50	**0.9693**	**0.9719**	3.5199	0.6683	× 1.8707	× 1.2293	+ 0.0041	+ 0.0091
70	0.9392	0.9353	3.1421	0.5645	× 2.095	× 1.4554	− 0.026	− 0.0275
90	0.5	0.3614	1.8498	0.4959	× **3.5598**	× **1.6567**	− 0.4652	− 0.6014

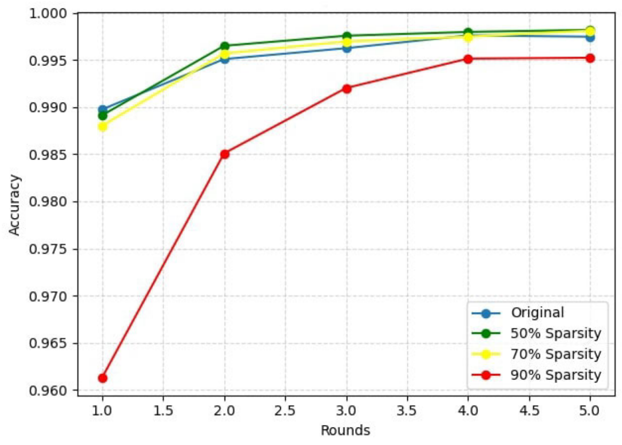

FIGURE 11. Convergence of the VGG-19 model in FL system with 5 rounds, and using the CIC-IDS-2017 dataset.

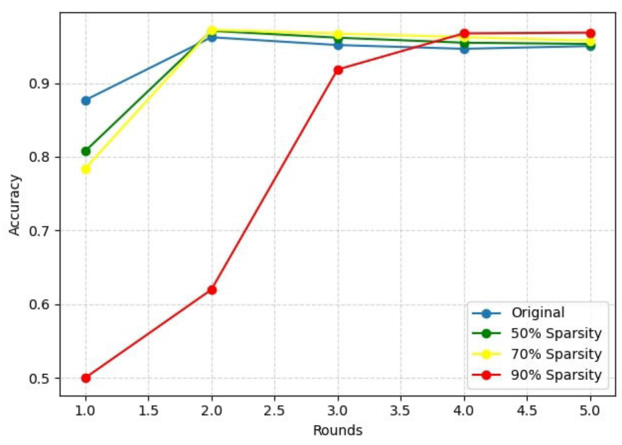

FIGURE 12. Convergence of the VGG-19 model in FL system with 5 rounds, and using the UNSW-NB15 dataset.

number of samples in the dataset, as shown in Figure 10. This explanation is substantiated by the results obtained from CIC-IDS-2017, which contains more than twice the number of samples compared to UNSW-NB15.

Next, we measure the impact of our FL model with 100 NIDS clients by alternating the feature extractors and datasets. Table 9 displays the obtained results with the CIC-IDS-2017 dataset and the VGG-19 feature extractor. Likewise, Table 10 summarizes the findings of our experiments with USTC-TFC2016 and ResNet-50 feature extractor. It is worth highlighting that we used ResNet-50 and VGG-19 in this evaluation to test the impact of pruning

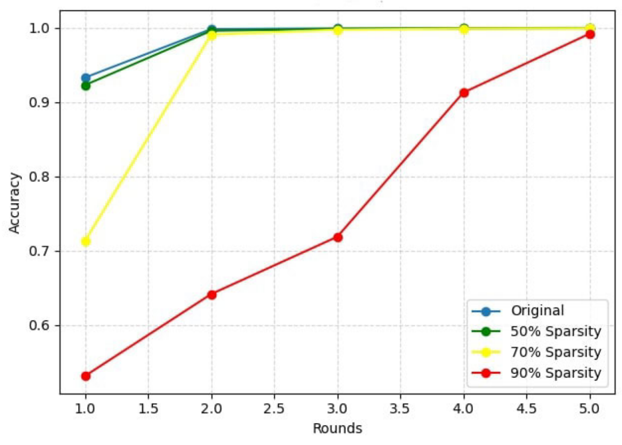

FIGURE 13. Convergence of the VGG-19 model in FL system with 5 rounds, and using the USTC-TFC2016 dataset.

on training and inference complexities while preserving the detection performance. Moreover, we proved the resilience of our approach using three datasets, namely CIC-IDS-2017, USTC-TFC2016, and UNSW-NB15, and three feature extractors, including ResNet50, ReNet-101, and VGG-19. In all cases, we demonstrated the efficiency of our framework in terms of detection performance and computation complexity optimization. As shown previously, there is an evident improvement in training and inference time complexities without dropping the performance even if we alter both the feature extractor and dataset. In addition, the successful implementation of our FL-based methodology across various participant sizes further supports the scalability, reliability, and applicability of our approach, highlighting its potential for deployment in real-world settings involving distributed environments where a large number of participants may be involved. It's worth mentioning that the results remain consistent regardless of variations in the number of clients, the feature extractor used, or the dataset employed.

Based on these results, our FL approach demonstrates the ability to efficiently prune neural networks without compromising performance. However, it's essential to fine-tune the feature extractor and sparsity level according to user preferences for a balance between speed and accuracy. The speedup achieved depends on various factors, including the chosen feature extractor, sparsity level, and dataset. Additionally, increasing the sparsity level can lead to improved training and inference times but may also result in a slight degradation of accuracy and F1-score. For example, if the user prioritizes both speed and accuracy, the model with a ResNet-101 feature extractor and a 70% sparsity level stands out with the highest F1-score (0.9993) among all models, and it also exhibits significant speed improvements in terms of training and inference (TA: $\times$ 2.03, IA: $\times$ 1.57). On the other hand, in scenarios where speed (inference time) takes precedence, the user might opt for a model featuring a VGG-19 extractor with a 90% sparsity level. This particular model achieves the highest TA ($\times$ 3.62) and IA ($\times$ 2.85) compared to all other models. However, it does come with a slightly lower accuracy (0.9973) and F1-score (0.995) when compared to alternative models.

To enhance our evaluation, we conducted additional experiments using the FedProx algorithm [48] on UNSW-NB15, USTC-TFC2016, and CIC-IDS-2017 datasets, employing the ResNet-50 architecture across configurations of 10 clients at various sparsity levels (0%, 50%, 70%, 90%). The results, presented in Table 11, reveal the performance of FedProx algorithm. Our findings consistently indicate that FedAvg (Table 6) either surpasses or matches the performance of FedProx in terms of accuracy and training time across all tested sparsity levels using 10 clients. This empirical evidence supports the effectiveness of FedAvg over more complex aggregation methods for our specific application, highlighting its suitability in scenarios requiring efficient and accurate federated learning.

It is noteworthy to state that the pruned models resulting from our approach contain a significant proportion of zero-valued parameters. This characteristic allows for more efficient compression and storage compared to dense models. Additionally, they can significantly reduce memory consumption by utilizing a sparse storage format that compresses repeated zeros, storing only the non-zero values and their corresponding locations [49], [50], [51]. This efficiency is particularly advantageous for IDS devices with limited memory capacity. While we did not directly conduct experiments on lower-end hardware, our methodology leverages a combination of model size reduction and computational complexity analysis to simulate and predict the impact on such devices. Firstly, the primary effect of pruning is to reduce the size of the model, which directly translates to lower memory requirements. By shrinking the model size, we inherently make the model more suitable for devices with limited RAM and storage capacity. This size reduction allows for faster loading times and less strain on the device's storage, which is a critical factor for limited-resource clients. Secondly, the reduction in the number of parameters and operations (FLOPs) as a result of pruning leads to a decrease in computational complexity. This directly impacts the CPU/GPU utilization during inference, leading to less power consumption and faster inference times, which are critical metrics for performance on limited-resource devices. By quantifying the reduction in computational complexity, we can simulate the expected improvements in inference speed and energy efficiency on such hardware. Moreover, the pruned models can help reduce the required bandwidth for transmitting model parameters over the network, which is advantageous in scenarios with limited or costly network connectivity.

Our Lightweight-Fed-NIDS approach demonstrates superior performance and efficiency compared to other state-of-the-art techniques in network intrusion detection. As shown in Table 12, our method achieves the highest F1-score (99.88%) and accuracy (99.89%) across multiple

TABLE 11. Performance of our approach using FedProx averaging algorithm on UNSW-NB15, USTC-TFC2016, and CIC-IDS-2017 datasets.

Dataset	Sparsity	Accuracy	F1-score	Tarining Time (s)	Inference Time (s)	TA	IA	Δ Acc	Δ F1-sc
UNSW-NB15	0	0.9439	0.8723	248.317	25.0086				
	50	0.9314	0.8783	156.684	16.3397	× 1.5848	× 1.5305	− 0.01255	+ 0.0060
	70	0.9273	0.8883	181.495	22.5792	× 1.3681	× 1.1075	− 0.01660	+ 0.0159
	90	0.9162	0.8891	137.355	18.1696	× 1.8078	× 1.3763	− 0.02768	+ 0.0168
USTC-TFC2016	0	0.9991	0.9992	65.9461	6.3971				
	50	0.9995	0.9995	41.2368	4.3263	× 1.5992	× 1.4787	+ 0.0004	+ 0.0004
	70	0.9994	0.9994	41.2042	4.1626	× 1.6005	× 1.5368	+ 0.0003	+ 0.0002
	90	0.9992	0.9993	30.0616	3.4685	× 2.1937	× 1.8444	+ 0.0001	+ 0.0001
CIC-IDS-2017	0	0.9960	0.9947	218.2783	21.8958				
	50	0.9960	0.9962	137.4222	14.6776	× 1.5884	× 1.4918	0	+ 0.0015
	70	0.9969	0.9952	137.2102	13.7927	× 1.5908	× 1.5875	+ 0.0009	+ 0.0006
	90	0.9969	0.9965	124.6203	17.8151	× 1.7515	× 1.2291	+ 0.0009	+ 0.0019

TABLE 12. Comparative analysis of our approach.

Method	Dataset	F1-score	Accuracy (%)	Key Features
MT-DNN-FL [9]	CIC-IDS-2017 ISCXVPN2016	–	98.14	- 1.7x faster than the centralized approach - Multi-Task Learning
AFL [26]	NSL-KDD	–	99.5	- Increased throughput - Partially Asynchronous communication
FedACNN [28]	NSL-KDD	88.97	99.12	Reduction of communication rounds
Our Method	**UNSW-NB15 USTC-TFC2016 CIC-IDS-2017**	**99.88%**	**99.89%**	**- data-free pruning technique - High detection performance - 3X faster in training and inference - Reduced the computational resource**

TABLE 13. Top obtained results across experiments using 10 clients.

Dataset	Unpruned				Pruned				
	Feature extractor	F1-score	Training time	Inference time	Feature extractor	F1-score	Training Time	Inference time	Sparsity
CIC-IDS	VGG-19	0.9989	111.1094	17.1264	VGG-19	0.9993	56.0593	12.4499	50%
UNSW	ResNet-50	0.947	85.9364	8.5626	VGG-19	0.9483	20.9921	3.8341	50%
USTC	VGG-19	0.9998	33.1345	4.4547	ResNet-50	0.9997	39.1443	4.1828	70%

datasets, including UNSW-NB15, USTC-TFC2016, and CIC-IDS-2017. This performance surpasses that of MT-DNN-FL [9], which reported 98.14% accuracy on CIC-IDS-2017, and AFL [26], which achieved 99.5% accuracy on the NSL-KDD dataset. Furthermore, our approach offers unique advantages through its data-free pruning technique, which contributes to a 3X speedup in both training and inference times, outperforming MT-DNN-FL's 1.7x speedup. While FedACNN [28] focuses on reducing communication rounds, our method addresses both computational efficiency and model performance. Notably, our approach is the only one among those compared that explicitly reduces computational resource requirements while maintaining high detection performance across a diverse range of datasets. This combination of high accuracy, computational efficiency, and resource optimization positions our Lightweight-Fed-NIDS as a robust and versatile solution for modern network intrusion detection challenges.

V. CONCLUSION AND FUTURE WORK

This paper presents a novel framework designed for NIDS. Lightweight-Fed-NIDS harnesses the advantages of federated learning and structured model pruning. It facilitates distributed learning across various local domains, updates all local models globally, and employs pruning techniques to decrease model size and complexity. The key advantage of our framework is the use of a non-iterative, structured, and data-free pruning technique called zero-shot pruning. This technique generates a pruning mask without using any data and applies it only once at the beginning of training.

Lightweight-Fed-NDIS achieves high detection performance in both centralized and FL settings with three well-known NIDS datasets, namely UNSW-NB2015 [22], USTC-TFC2016 Wang et al. [23] and CICIDS-2017 [21]. For example, the model built with the ResNet-50 feature extractor and 70% pruning rate scored 99% accuracy and F1-score in both centralized and distributed cases with

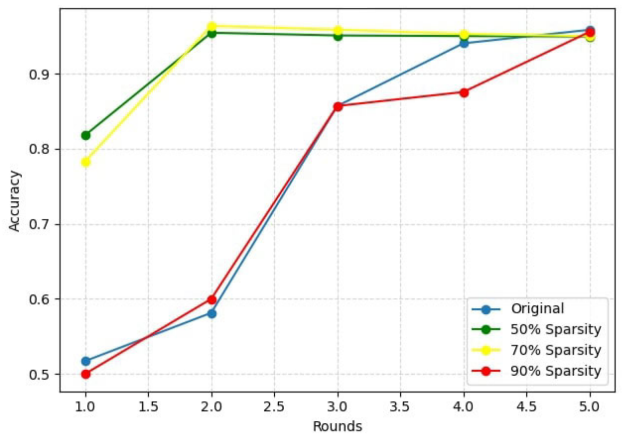

FIGURE 14. **Convergence of the ResNet-50 model in FL system with 5 rounds, and using UNSW-NB15.**

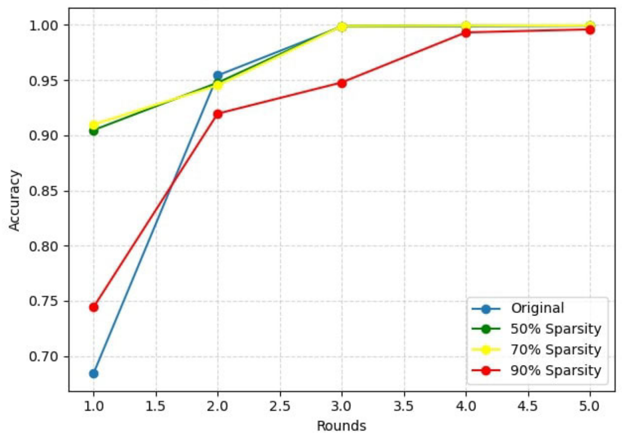

FIGURE 15. **Convergence of the ResNet-50 model in FL system with 5 rounds using USTC-TFC2016.**

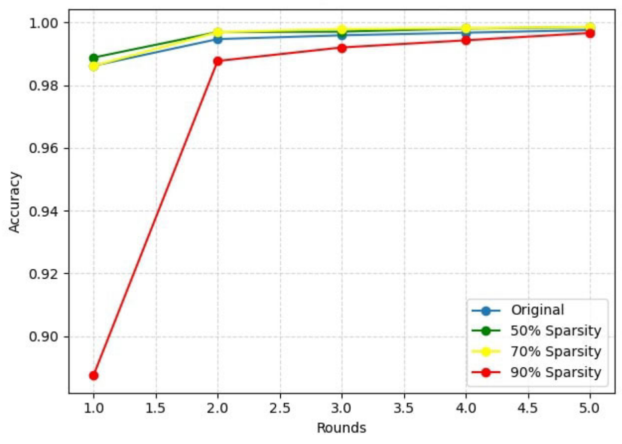

FIGURE 16. **Convergence of the ResNet-50 model in FL system with 5 rounds using CIC-IDS-2017.**

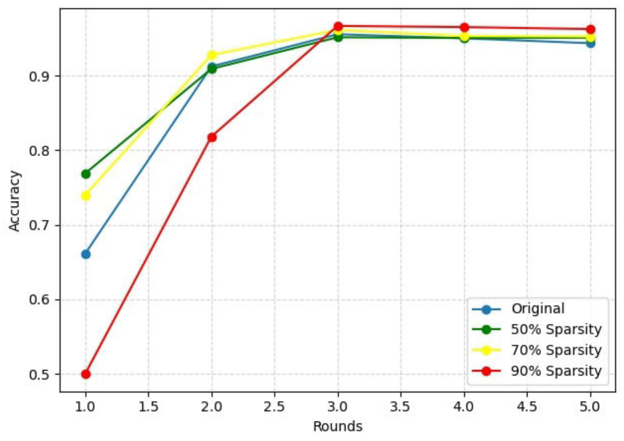

FIGURE 17. **Convergence of the ResNet-101 model in FL system with 5 rounds using UNSW-NB15.**

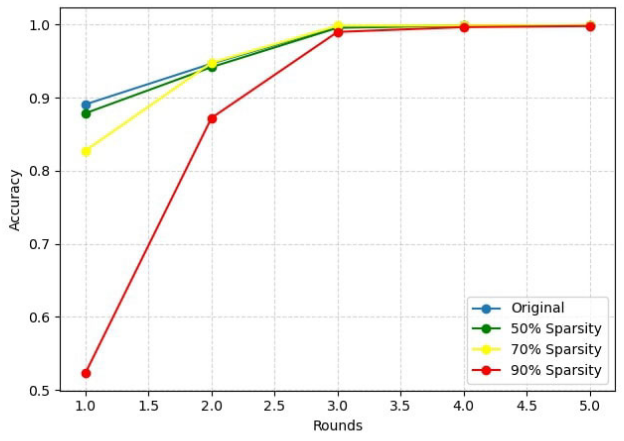

FIGURE 18. **Convergence of the ResNet-101 model in FL system with 5 rounds using USTC-TFC2016.**

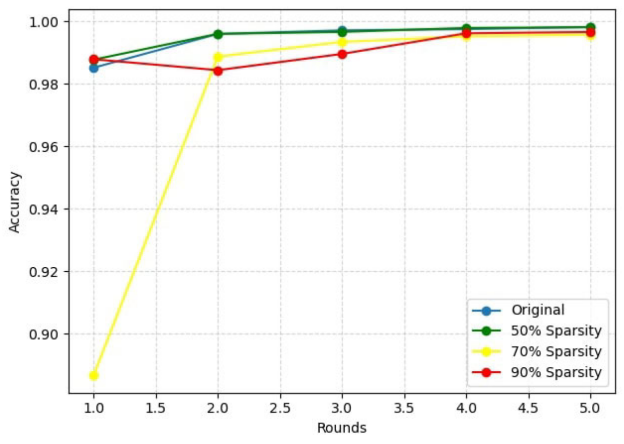

FIGURE 19. **Convergence of the ResNet-101 model in FL system with 5 rounds using CICIDS2017.**

the CIC-IDS-2017 dataset. This resulted in improving the training and inference time by up to 3X faster and reducing the computational resource consumption compared to the unpruned model (baseline). This makes our system

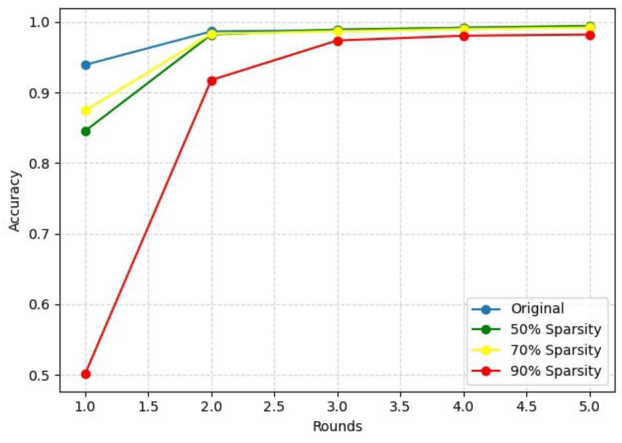

FIGURE 20. Convergence of the ResNet-50 model in FL system with 5 rounds using CIC-IDS-2017 and involving 50 users.

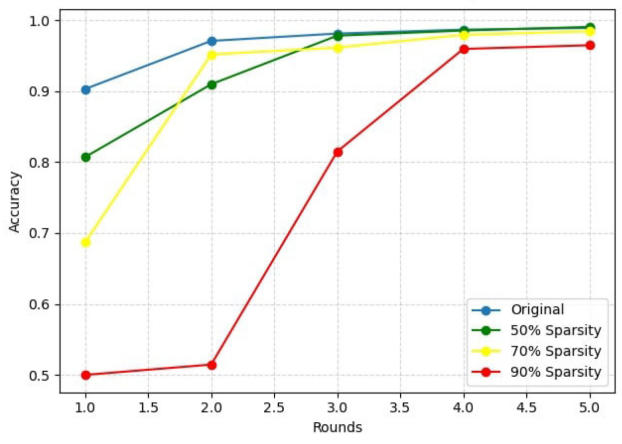

FIGURE 21. Convergence of the ResNet-50 model in FL system with 5 rounds using CIC-IDS-2017 and involving 100 users.

practical for limited-resource environments. Furthermore, we measured the performance of our method with various scenarios and showed that it is adaptable to different settings of FL, different datasets, and feature extractors.

However, we should also acknowledge the existing technical challenges and difficulties of applying FL techniques, such as non-independent and identically distributed (Non-IID) datasets and privacy issues. Therefore, in future work, we plan to address the problem of Non-IID datasets, which can affect the performance and convergence of FL, and tackle some privacy issues such as poisoning attacks, where malicious agents can inject false data or labels to compromise the global model, and honest-but-curious servers, where the central authority can infer sensitive information from the local models. Additionally, we aim to explore the applicability of our approach to specific network environments with unique security requirements, including Industrial Control Systems (ICS). This future work will further enhance the versatility of Lightweight-Fed-NIDS across diverse network infrastructures, from general IT networks to specialized operational technology environments. By adapting our framework to the stringent real-time and reliability demands of ICS, we can potentially address critical infrastructure security challenges while maintaining the privacy and efficiency benefits of our federated learning approach.

In conclusion, Lightweight-Fed-NIDS offers a promising approach to network intrusion detection that balances high performance with computational efficiency. By leveraging federated learning and model pruning, our framework addresses key challenges in modern network security, paving the way for more robust and adaptable NIDS solutions in various network environments.

APPENDIX

In this section, we present supplementary experiments focused on assessing the convergence of models utilizing ResNet-50 and ResNet-100 feature extractors, in conjunction with the UNSW-NB2015, USTC-TFC2016, and CICIDS-2017 datasets.

REFERENCES

[1] Y. Hu, A. Yang, H. Li, Y. Sun, and L. Sun, "A survey of intrusion detection on industrial control systems," *Int. J. Distrib. Sensor Netw.*, vol. 14, no. 8, Aug. 2018, Art. no. 155014771879461, doi: 10.1177/1550147718794615.

[2] B. Li, Y. Wu, J. Song, R. Lu, T. Li, and L. Zhao, "DeepFed: Federated deep learning for intrusion detection in industrial cyber–physical systems," *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5615–5624, Aug. 2021, doi: 10.1109/TII.2020.3023430.

[3] T. T. Huong, T. P. Bac, D. M. Long, T. D. Luong, N. M. Dan, L. A. Quang, L. T. Cong, B. D. Thang, and K. P. Tran, "Detecting cyberattacks using anomaly detection in industrial control systems: A federated learning approach," *Comput. Ind.*, vol. 132, Nov. 2021, Art. no. 103509, doi: 10.1016/j.compind.2021.103509.

[4] H. T. Truong, B. P. Ta, Q. A. Le, D. M. Nguyen, C. T. Le, H. X. Nguyen, H. T. Do, H. T. Nguyen, and K. P. Tran, "Light-weight federated learning-based anomaly detection for time-series data in industrial control systems," *Comput. Ind.*, vol. 140, Sep. 2022, Art. no. 103692, doi: 10.1016/j.compind.2022.103692.

[5] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, p. 20, Dec. 2019, doi: 10.1186/s42400-019-0038-7.

[6] R. Mitchell and I.-R. Chen, "A survey of intrusion detection techniques for cyber-physical systems," *ACM Comput. Surv.*, vol. 46, no. 4, pp. 1–29, Mar. 2014, doi: 10.1145/2542049.

[7] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2016, *arXiv:1610.05492*.

[8] J.-S. Lee, Y.-C. Chen, C.-J. Chew, C.-L. Chen, T.-N. Huynh, and C.-W. Kuo, "CoNN-IDS: Intrusion detection system based on collaborative neural networks and agile training," *Comput. Secur.*, vol. 122, Nov. 2022, Art. no. 102908, doi: 10.1016/j.cose.2022.102908.

[9] Y. Zhao, J. Chen, D. Wu, J. Teng, and S. Yu, "Multi-task network anomaly detection using federated learning," in *Proc. 10th Int. Symp. Inf. Commun. Technol. (SoICT)*, New York, NY, USA, 2019, pp. 273–279, doi: 10.1145/3368926.3369705.

[10] S. Agrawal, S. Sarkar, O. Aouedi, G. Yenduri, K. Piamrat, M. Alazab, S. Bhattacharya, P. K. R. Maddikunta, and T. R. Gadekallu, "Federated learning for intrusion detection system: Concepts, challenges and future directions," *Comput. Commun.*, vol. 195, pp. 346–361, Nov. 2022, doi: 10.1016/j.comcom.2022.09.012.

[11] Z. Wang, Z. Li, D. He, and S. Chan, "A lightweight approach for network intrusion detection in industrial cyber-physical systems based on knowledge distillation and deep metric learning," *Expert Syst. Appl.*, vol. 206, Nov. 2022, Art. no. 117671, doi: 10.1016/j.eswa.2022.117671.

[12] R. A. Sater and A. B. Hamza, "A federated learning approach to anomaly detection in smart buildings," *ACM Trans. Internet Things*, vol. 2, no. 4, pp. 1–23, Nov. 2021, doi: 10.1145/3467981.

[13] D. Wu, Y. Deng, and M. Li, "FL-MGVN: Federated learning for anomaly detection using mixed Gaussian variational self-encoding network," *Inf. Process. Manage.*, vol. 59, no. 2, Mar. 2022, Art. no. 102839, doi: 10.1016/j.ipm.2021.102839.

[14] S. Anwar, K. Hwang, and W. Sung, "Structured pruning of deep convolutional neural networks," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 13, no. 3, pp. 1–18, Jul. 2017, doi: 10.1145/3005348.

[15] G. Fang, X. Ma, M. Song, M. Bi Mi, and X. Wang, "DepGraph: Towards any structural pruning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Vancouver, BC, Canada, Jun. 2023, pp. 16091–16101, doi: 10.1109/CVPR52729.2023.01544.

[16] N. Liu, X. Ma, Z. Xu, Y. Wang, J. Tang, and J. Ye, "AutoCompress: An automatic DNN structured pruning framework for ultra-high compression rates," in *Proc. 34th AAAI Conf. Artif. Intell. (AAAI) 32nd Innov. Appl. Artif. Intell. Conf. (IAAI) 10th AAAI Symp. Educ. Adv. Artif. Intell. (EAAI)*, Apr. 2020, vol. 34, no. 4, pp. 4876–4883. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/5924

[17] M. Xia, Z. Zhong, and D. Chen, "Structured pruning learns compact and accurate models," in *Proc. 60th Annu. Meeting Assoc. Comput. Linguistics (ACL)*, Dublin, Ireland, S. Muresan, P. Nakov, and A. Villavicencio, Eds., Stroudsburg, PA, USA: Association for Computational Linguistics, May 2022, pp. 1513–1528, doi: 10.18653/v1/2022.acl-long.107.

[18] A. Bragagnolo and C. A. Barbano, "Simplify: A Python library for optimizing pruned neural networks," *SoftwareX*, vol. 17, Jan. 2022, Art. no. 100907, doi: 10.1016/j.softx.2021.100907.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015, *arXiv:1512.03385*.

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, San Diego, CA, USA, Y. Bengio and Y. LeCun, Eds., May 2015, pp. 1–14.

[21] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy*, Funchal, Portugal, P. Mori, S. Furnell, and O. Camp, Eds., Setúbal, Portugal: SciTePress, Jan. 2018, pp. 108–116, doi: 10.5220/0006639801080116.

[22] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS)*, Canberra, ACT, Australia, Nov. 2015, pp. 1–6, doi: 10.1109/MILCIS.2015.7348942.

[23] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2017, pp. 712–717.

[24] A. H. Lashkari, G. D. Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of tor traffic using time based features," in *Proc. 3rd Int. Conf. Inf. Syst. Secur. Privacy*, Porto, Portugal, P. Mori, S. Furnell, and O. Camp, Eds., Setúbal, Portugal: SciTePress, Feb. 2017, pp. 253–262, doi: 10.5220/0006105602530262.

[25] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and VPN traffic using time-related features," in *Proc. 2nd Int. Conf. Inf. Syst. Secur. Privacy (ICISSP)*, Rome, Italy. Setúbal, Portugal: SciTePress, Feb. 2016, pp. 407–414, doi: 10.5220/0005740704070414.

[26] S. Agrawal, A. Chowdhuri, S. Sarkar, R. Selvanambi, and T. R. Gadekallu, "Temporal weighted averaging for asynchronous federated intrusion detection systems," *Comput. Intell. Neurosci.*, vol. 2021, no. 1, pp. 1–10, Jan. 2021.

[27] Z. Zhang, Y. Zhang, D. Guo, L. Yao, and Z. Li, "SecFedNIDS: Robust defense for poisoning attack against federated learning-based network intrusion detection system," *Future Gener. Comput. Syst.*, vol. 134, pp. 154–169, Sep. 2022, doi: 10.1016/j.future.2022.04.010.

[28] D. Man, F. Zeng, W. Yang, M. Yu, J. Lv, and Y. Wang, "Intelligent intrusion detection based on federated learning for edge-assisted Internet of Things," *Secur. Commun. Netw.*, vol. 2021, pp. 1–11, Oct. 2021.

[29] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, Barcelona, Spain, D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, Eds., Dec. 2016, pp. 2074–2082. [Online]. Available: https://proceedings.neurips.cc/paper/2016/hash/41bfd20a38bb1b0bec75acf0845530a7-Abstract.html

[30] C. Louizos, M. Welling, and D. P. Kingma, "Learning sparse neural networks through L_0 regularization," 2017, *arXiv:1712.01312*.

[31] E. Tartaglione, A. Bragagnolo, F. Odierna, A. Fiandrotti, and M. Grangetto, "SeReNe: Sensitivity-based regularization of neurons for structured sparsity in neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 12, pp. 7237–7250, Dec. 2022.

[32] K. Ullrich, E. Meeds, and M. Welling, "Soft weight-sharing for neural network compression," in *Proc. 5th Int. Conf. Learn. Represent. (ICLR)*, Toulon, France, Apr. 2017, pp. 1–16. [Online]. Available: https://openreview.net/forum?id=HJGwcKclx

[33] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *Proc. 7th Int. Conf. Learn. Represent. (ICLR)*, New Orleans, LA, USA, May 2019, pp. 1–42. [Online]. Available: https://openreview.net/forum?id=rJl-b3RcF7

[34] J. Su, Y. Chen, T. Cai, T. Wu, R. Gao, L. Wang, and J. D. Lee, "Sanity-checking pruning methods: Random tickets can win the jackpot," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., Dec. 2020, pp. 1–12. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/eae27d77ca20db309e056e3d2dcd7d69-Abstract.html

[35] H. Tanaka, D. Kunin, D. L. K. Yamins, and S. Ganguli, "Pruning neural networks without any data by iteratively conserving synaptic flow," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., Dec. 2020, pp. 1–13. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/46a4378f835dc8040c8057beb6a2da52-Abstract.html

[36] Y. Cai, W. Hua, H. Chen, G. E. Suh, C. De Sa, and Z. Zhang, "Structured pruning is all you need for pruning CNNs at initialization," 2022, *arXiv:2203.02549*.

[37] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Salt Lake City, UT, USA, Jun. 2018, pp. 4510–4520. [Online]. Available: http://openaccess.thecvf.com/content_cvpr_2018/html/Sandler_MobileNetV2_Inverted_Residuals_CVPR_2018_paper.html

[38] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. Mach. Learn. (ICML)*, in Proceedings of Machine Learning Research, vol. 97, Long Beach, CA, USA, K. Chaudhuri and R. Salakhutdinov, Eds., 2019, pp. 6105–6114. [Online]. Available: http://proceedings.mlr.press/v97/tan19a.html

[39] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," 2012, *arXiv:1207.0580*.

[40] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Miami, FL, USA, Jun. 2009, pp. 248–255, doi: 10.1109/CVPR.2009.5206848.

[41] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1026–1034.

[42] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Mach. Learn. (ICML)*, vol. 37, Lille, France, F. R. Bach and D. M. Blei, Eds., Jul. 2015, pp. 448–456. [Online]. Available: http://proceedings.mlr.press/v37/ioffe15.html

[43] Z. Aouini and A. Pekar, "Nfstream: A flexible network data analysis framework," *Comput. Netw.*, vol. 204, Feb. 2022, Art. no. 108719. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128621005739

[44] H. Alami, M. J. Idrissi, A. E. Mahdaouy, A. Bouayad, Z. Yartaoui, and I. Berrada, "Investigating domain adaptation for network intrusion detection," in *Proc. 10th Int. Conf. Wireless Netw. Mobile Commun. (WINCOM)*, Oct. 2023, pp. 1–7.

[45] P. Sun, P. Liu, Q. Li, C. Liu, X. Lu, R. Hao, and J. Chen, "DL-IDS: Extracting features using CNN-LSTM hybrid network for intrusion detection system," *Secur. Commun. Netw.*, vol. 2020, Aug. 2020, Art. no. 8890306, doi: 10.1155/2020/8890306.

[46] X. Liu, Z. Tang, and B. Yang, "Predicting network attacks with CNN by constructing images from NetFlow data," in *Proc. IEEE IEEE 5th Intl Conf. Big Data Secur. Cloud (BigDataSecurity) Intl Conf. High Perform. Smart Comput., (HPSC) IEEE Intl Conf. Intell. Data Secur. (IDS)*, Washington, DC, USA, May 2019, pp. 61–66, doi: 10.1109/BigDataSecurity-HPSC-IDS.2019.00022.

[47] V. Pham, E. Seo, and T.-M. Chung, "Lightweight convolutional neural network based intrusion detection system," *J. Commun.*, vol. 15, no. 11, pp. 808–817, 2020, doi: 10.12720/jcm.15.11.808-817.

[48] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. Mach. Learn. Syst. (MLSys)*, Austin, TX, USA, I. S. Dhillon, D. S. Papailiopoulos, and V. Sze, Eds., Mar. 2020, pp. 1–22. [Online]. Available: https://proceedings.mlsys.org/book/316.pdf

[49] E. Im and K. A. Yelick, "Optimizing sparse matrix vector multiplication on SMP," in *Proc. 9th SIAM Conf. Parallel Process. Sci. Comput. (PPSC)*, San Antonio, TX, USA, Mar. 1999, pp. 1–9.

[50] J. Dongarraxz, A. Lumsdaine, X. Niu, R. Pozoz, and K. Remingtonx, "A sparse matrix library in C++ for high performance architectures," in *Proc. 2nd Object Oriented Numerics Conf.*, 1994, pp. 214–218.

[51] A. Ekambaram and E. Montagne, "An alternative compressed storage format for sparse matrices," in *Proc. 18th Int. Symp. Comput. Inf. Sci. (ISCIS)*, in Lecture Notes in Computer Science, vol. 2869, Antalya, Turkey, A. Yazici and C. Sener, Eds., Cham, Switzerland: Springer, Nov. 2003, pp. 196–203, doi: 10.1007/978-3-540-39737-3_25.

ABDELHAK BOUAYAD received the M.S. degree in big data analytics and smart systems from Sidi Mohamed Ben Abdellah University, Fes, Morocco, in 2018. He is currently pursuing the Ph.D. degree in data science, networking and algorithmic thinking with Mohammed VI Polytechnic University (UM6P), Morocco. His research interests include privacy-preserving machine learning, network intrusion detection systems, and federated learning.

HAMZA ALAMI received the Ph.D. degree in computer science from Ibn Tofail University, Morocco, in 2021. He is currently an Assistant Professor with Sidi Mohammed Ben Abdellah University (USMBA), Fes, Morocco. His research interests include natural language processing, distributed learning, network intrusion detection, and information extraction.

MERYEM JANATI IDRISSI received the M.S. degree in big data analytics and smart systems from Sidi Mohamed Ben Abdellah University, Fes, Morocco, in 2018, and the Ph.D. degree in data science, networking and algorithmic thinking from Mohammed VI Polytechnic University (UM6P), Morocco. Her current research interests include intrusion detection, federated learning, and privacy preserving.

ISMAIL BERRADA received the Ph.D. degree in computer science from the University of Bordeaux 1, France, in 2005. He is currently an Associate Professor in computer science with Mohammed VI Polytechnic University (UM6P), Morocco. He worked for five years as an Assistant Professor at the University of La Rochelle, France, and ten years at Sidi Mohammed Ben Abdellah University (USMBA), Fes, Morocco. His research interests include artificial intelligence (AI) applications in multiple domains, such as cognitive radio networks, radio resource management, vehicular ad hoc networks, road safety, software testing, and verification.