FIDSUS: Federated Intrusion Detection for Securing UAV Swarms in Smart Aerial Computing

Jiangtao Deng, Wei Wang, *Member, IEEE*, Lina Wang, Ali Kashif Bashir, *Senior Member, IEEE*, Thippa Reddy Gadekallu, *Senior Member, IEEE*, Hailin Feng, *Member, IEEE*, Meilei Lv, and Kai Fang, *Member, IEEE*

***Abstract*—The dynamic environment of UAV swarms in forest management is characterized by communication instability, heterogeneous nodes, and frequent topology changes due to challenging terrain. These systems are vulnerable to network attacks, requiring advanced intrusion detection technologies. Traditional methods struggle with rapid changes due to data privacy concerns and centralized computational limits, while existing Federated Learning (FL) algorithms lack robustness against client heterogeneity and dynamic data distribution, especially in complex forest environments. To address these challenges, we propose Federated Intrusion Detection for Securing UAV Swarms (FIDSUS). FIDSUS improves intrusion detection systems by leveraging collaborative sensing among UAVs, enabling better monitoring and response to security threats in forestry. By quantifying the similarity between UAVs' local feature extractors through an affinity matrix, FIDSUS guides the aggregation of feature extractors, improving detection capabilities. It also uses AI-driven aerial and distributed computing to enhance data processing efficiency and decision-making speed. The framework addresses data heterogeneity by cross-round feature fusion, improving detection in dynamic environments. Experimental results on the NSL-KDD and UNSW-NB15 datasets show that FIDSUS outperforms existing FL methods with a 4% to 34% accuracy improvement. FIDSUS shows robustness and accuracy in dynamic environments, providing an effective solution for securing UAV swarms in forestry.**

This work was partly supported by the National Natural Science Foundation of China under grant no.62403433; The Zhejiang Provincial Natural Science Foundation of China under grant no. LQ23F020001; The Zhejiang Major Water Conservancy Science and Technology Project under grant no. RA2201.

Code for this paper is available at https://github.com/ryota7777/FIDSUS.

Jiangtao Deng is with the College of Mathematics and Computer Science, Zhejiang A&F University, Hangzhou, 311300, China, and with the College of Electrical and Information Engineering, Quzhou University, Quzhou, 324000, China. (Email: dengjiangtao07@gmail.com)

Wei Wang is with the Guangdong-Hong Kong-Macao Joint Laboratory for Emotion Intelligence and Pervasive Computing, Artificial Intelligence Research Institute, Shenzhen MSU-BIT University, Shenzhen 518172, China, and are also with the School of Medical Technology, Beijing Institute of Technology, Beijing 100081, China (Email: ehomewang@ieee.org)

Lina Wang, Hailin Feng, and Kai Fang are with the College of Mathematics and Computer Science, Zhejiang A&F University, Hangzhou, 311300, China. (Email: Linawangzafu@gmail.com, hlfeng@zafu.edu.cn, and Kaifang@ieee.org)

Thippa Reddy Gadekallu is with The College of Mathematics and Computer Science, Zhejiang A&F University, Hangzhou 311300, China as well as with the Division of Research and Development, Lovely Professional University, Phagwara, India and with the Center of Research Impact and Outcome, Chitkara University, Rajpura, 140401, Punjab, India (Email: Thippareddy@ieee.org)

Ali Kashif Bashir is with the Department of Computing and Mathematics, Manchester Metropolitan University, M15 6BX Manchester, U.K., and also with the Centre for Research Impact and Outcome, Chitkara University Institute of Engineering and Technology, Chitkara University, Rajpura, Punjab 140401, India (Email: dr.alikashif.b@ieee.org).

Meilei Lv is with the College of Electrical and Information Engineering, Quzhou University, Quzhou, 324000, China (Email: 37014@qzc.edu.cn)

Corresponding authors: Meilei Lv, and Kai Fang

***Index Terms*—Federated learning, UAV swarms, intrusion detection, cybersecurity, edge computing.**

I. Introduction

FORESTS play a vital role in maintaining biodiversity, regulating climate, and providing resources for human communities. However, the challenges of monitoring large, often inaccessible areas, combined with the need for timely responses to issues such as forest fires, illegal logging, and wildlife protection, make traditional management methods increasingly inadequate. Effective monitoring requires real-time data collection, analysis, and decision-making to address environmental threats swiftly and accurately. By leveraging flexibility, mobility, and the ability to cover large areas quickly, UAV swarms can provide crucial support in fields such as forestry management, emergency rescue, and smart city development [1]–[3]. Equipped with diverse sensors for environmental monitoring, UAVs can capture high-resolution data on vegetation health, forest density, and the presence of wildlife, even in difficult-to-reach areas. Their ability to operate in remote forests, where communication infrastructure is often sparse, further enhances their value. Moreover, AI-driven aerial computing empowers UAV swarms with intelligent decision-making and autonomous learning capabilities, allowing them to adaptively adjust network topology and resource allocation in response to environmental changes [4]. This combination of UAV collaborative sensing and AI-driven aerial computing enhances their ability to perform complex tasks in forest management, such as wildlife monitoring, fire detection, and ecosystem analysis [5].

Studies have shown that UAV swarms are capable of autonomous navigation and coordination in complex environments while maintaining efficient communication and avoiding collisions, which promoted the application of UAV swarms in complex forest environments [6]. By using multiple UAVs working collaboratively, the time required for detecting and extinguishing fires can be significantly reduced, effectively limiting the damage to forest ecosystems and biodiversity [7]. In forest resource management, [8] used multiple UAVs to measure tree parameters, such as trunk diameter and height, at a lower cost and in a shorter amount of time. Sangaiah et al. utilized UAVs to enhance the detection and localization of paddy leaf diseases, achieving improved accuracy through their proposed UAV T-YOLO-Rice network [9] and R-UAV-Net [10], which integrates advanced feature extraction blocks and attention mechanisms for more precise disease detection.

Furthermore, [11] implemented collaborative synchronization and mapping technologies using multiple UAVs under the forest canopy, which have also been applied in search and rescue operations.

However, the use of UAV swarms in forestry applications also introduces significant challenges. While the collaborative nature of UAV swarms enhances their capability to monitor vast forested regions, it also exposes the system to new security risks. The openness and connectivity of these systems make them vulnerable to malicious attacks, such as data theft or system manipulation, which can compromise both the collected data and the overall mission [12]–[14]. Given the dynamic nature of the forest environment and the complexities of maintaining secure communication in such settings, developing robust Intrusion Detection Systems (IDS) to protect UAV swarms is crucial [15].

Deep learning models can process large volumes of data and extract complex patterns, making them generally more effective than traditional rule based IDSs [16]. However, traditional IDS often rely on centralized architectures, which aggregate all data to a central server for analysis and processing. This centralized approach encounters significant challenges in UAV swarm applications. It must manage large volumes of distributed sensor data, which are frequently real-time, sensitive, and heterogeneous. To process this data, centralized architectures require the constant transmission of massive amounts of information from various nodes to a central server. This not only results in high communication overhead but also heightens the risk of privacy breaches during data transfer. Furthermore, centralized data processing can lead to delays, compromising real-time performance and overall system efficiency [17]. Therefore, traditional centralized intrusion detection solutions have significant limitations in UAV swarm applications [18]. They fall short of meeting the high security and efficiency demands required by modern UAV swarms. In this context, exploring new and more suitable IDS for distributed environments is crucial. This approach will help effectively address the complex security challenges faced by UAV swarms and ensure their reliable operation across various forestry application scenarios.

FL, as an emerging machine learning paradigm, offers a promising solution, especially in scenarios with high data privacy and security requirements. It enables multiple distributed devices to collaboratively train a shared model without exchanging raw data. This approach protects data privacy while leveraging data from each node to optimize the model [19]. Using FL, UAV swarm nodes can independently train models on their local data. They then send the resulting model updates to a central server for aggregation [20]. This process updates the global model, enabling the entire system to continuously learn and adapt to new data and threats. In forestry management, where UAVs collect data from remote and isolated regions, this decentralized approach significantly enhances the ability to analyze environmental patterns and detect potential threats such as unauthorized access. This method not only effectively protects data privacy but also improves the flexibility and efficiency of intrusion detection. By deploying intrusion detection models within the FL framework, real-time responses to security threats in distributed environments can be achieved. This approach ensures the stability and security of UAV swarms. However, existing FL algorithms still face numerous challenges when applied to highly dynamic UAV swarm environments. The frequent joining and leaving of UAV swarm nodes lead to dynamic client changes, which in turn cause instability in the feature space [21]. This instability not only reduces the consistency of model training but also impacts the accuracy, response speed, and reliability. Forests, with their unpredictable terrain and environmental conditions, further exacerbate these challenges, as the swarm must continuously adapt to new threats and network disruptions. Additionally, the complex topology and unreliable nodes present additional challenges for FL [22]. Frequent communication disruptions and non-independent and identically distributed (non-IID) data distribution complicate model training and updating [23]. In real-world scenarios, network traffic data often exhibit highly imbalanced class distributions, where certain types of traffic, such as attacks, are much less frequent than normal traffic. When such imbalanced datasets are partitioned using a Dirichlet distribution, the resulting partitions can worsen the imbalance, leading to even more extreme distributions [24], [25]. This increased imbalance can significantly impact model performance, making it harder for the model to detect the less frequent class effectively. Most existing Federated Learning (FL) algorithms are typically validated on balanced datasets, so it is unclear whether they can effectively handle such imbalanced class distributions. These challenges severely hinder the practical application of FL in UAV swarm intrusion detection. Therefore, there is an urgent need to design a federated intrusion detection framework that effectively addresses dynamic client changes. This framework should ensure the stability and accuracy of intrusion detection while optimizing communication and computational overhead during training.

To address intrusion detection in highly dynamic environments, we propose a federated learning-based intrusion detection framework, FIDSUS. This framework enables each UAV device to collaboratively build a global intrusion detection model without transmitting sensitive local data to a central server. FIDSUS decouples the client's local model into a feature extractor and a classifier. During the feature extraction phase, FIDSUS constructs an affinity matrix to quantify the similarity between feature extractors of different clients. This allows FIDSUS to prioritize the aggregation of feature extractors from clients with high similarity, thereby improving the model's adaptability to local data features. Even in rapidly changing UAV swarm environments with shifting terminals and network topologies, FIDSUS maintains effective intrusion detection capabilities. During training, clients employ Maximum Mean Discrepancy (MMD) [26] to quantify the divergence between feature representations from different training rounds. These discrepancies are then used to weight the aggregation of local feature representations, which are subsequently uploaded to the server. The server trains a global classifier using these aggregated feature representations. This approach allows FIDSUS to learn more generalizable feature representations, thereby enhancing its generalization and adaptability. Through these strategies, FIDSUS improves

the precision and reliability of intrusion detection models in practical applications, offering robust security guarantees for highly dynamic UAV swarm environments. The main contributions of this paper are as follows:

1) We propose a dynamic knowledge-sharing mechanism for federated learning in UAV swarms, where clients iteratively update an affinity matrix. This allows clients with similar data distributions to share knowledge more effectively, improving feature extractor personalization and enhancing model adaptability and robustness in dynamic UAV environments.
2) We use Maximum Mean Discrepancy (MMD) to measure distributional differences between feature representations across rounds, addressing the catastrophic forgetting problem in federated learning. This enables better weighting and aggregation of new feature representations, improving the model's generalization and robustness across heterogeneous data.
3) Experimental results on the UNSW-NB15 and NSL-KDD datasets show that FIDSUS outperforms existing methods in handling complex, non-IID network traffic data. Additionally, it demonstrates superior robustness in dynamic UAV scenarios while maintaining reasonable time costs.

The remainder of this paper is organized as follows: Section II reviews related work, Section III discusses the methodology, Section IV presents the experiments and analysis, and Section V concludes the paper.

II. Related Work

A. UAV Swarm Security

UAV swarm technology has advanced rapidly in recent years, finding widespread use in emergency rescue, environmental monitoring, and communication infrastructure construction. Despite their flexibility and efficiency, UAV swarms are also susceptible to numerous security challenges. Operating in open and dynamic environments makes UAVs vulnerable to threats such as data theft, system tampering, and physical hijacking. These risks not only affect the functioning of individual UAVs but also jeopardize the entire swarm due to their interconnected nature. To address these security concerns, researchers have proposed various solutions. Li et al. [27] designed an efficient and secure communication protocol. This protocol combines SM4 encryption, BLS signature, and Merkle hash tree technology. It enhances network security and reduces key management costs. Xie et al. [28] proposed a blockchain-based multi-UAV task management scheme (B-UAVM). This scheme accelerates the consensus process through a three-layer blockchain structure and an improved Byzantine fault tolerance mechanism (IPBFT). It ensures the security of task data and entity information. Ye et al. [29] introduced a robust secure communication scheme based on intelligent reflecting surfaces (IRS). This scheme optimizes UAV transmission power, trajectory, and IRS phase shifters. It enhances secure communication between UAVs under imperfect channel conditions. While these solutions have enhanced UAV swarm security, significant limitations remain. Many struggle to adapt to the highly dynamic network environment typical of UAV swarms, where frequent changes in topology and the constant joining or departure of nodes complicate the maintenance of system security. Furthermore, the lack of support for heterogeneous networks hampers the protection of communications across different types of devices. Although encryption and blockchain technologies improve security, they also consume considerable resources, limiting their application in resource-constrained environments. As the size of UAV swarms increases, these approaches often fail to scale effectively, making it difficult to meet the security demands of large-scale cooperative operations.

B. Intrusion Detection Systems

IDS are widely employed to counter various security threats in distributed systems. Traditional centralized IDS work well with centralized data by collecting and analyzing data from a single location. However, these systems face significant limitations in distributed environments, where data is spread across multiple nodes. Key challenges include high communication overhead, insufficient real-time response, and risks to data privacy [30]. Centralized IDS often require large volumes of data to be transmitted from distributed nodes to a central server, which increases network load and can lead to delays or data loss. Furthermore, concentrating all data in one location makes the central server a prime target for attacks, thereby exposing sensitive information and undermining privacy. Additionally, centralized systems struggle with maintaining accuracy and efficiency when processing heterogeneous data in real-time due to the vast and dynamic nature of distributed networks. To overcome these limitations, researchers have turned to Distributed Intrusion Detection Systems (DIDS), which use decentralized architectures to improve security and performance. DIDS systems process data locally on each node, reducing the need for excessive communication and enhancing system scalability. For instance, Parra et al. [31] proposed a distributed deep learning framework for detecting and mitigating phishing, Distributed Denial of Service (DDoS), and Botnet attacks in IoT devices. This framework includes Distributed Convolutional Neural Networks (DCNN) embedded in devices. It also features a cloud-based Long Short-Term Memory (LSTM) network model. These components work collaboratively to detect attacks at both the device and backend levels, thus enhancing overall system security. Zhao et al. [32] introduced a low-load DIDS task scheduling method based on Q-Learning. This method dynamically adjusts scheduling strategies to balance system load and packet loss rate. Intelligent task scheduling allows the system to better respond to load changes. It maintains low latency and efficient data processing capabilities. Mousa'B et al. [33] proposed an explainable ensemble deep learning-based intrusion detection system for Industrial Internet of Things (IIoT) networks, utilizing SHAP and LIME methods to enhance transparency and robustness while reducing false positives. Tlili et al. [34] proposed an enhanced Distributed Intrusion Detection System (E-DIDS) specifically designed for UAVs. By deploying multiple interconnected IDS units,

the system improves security, optimizes attack detection performance, and reduces resource consumption. While DIDS mitigates some limitations of centralized IDS, it still faces scalability and robustness issues when dealing with large, dynamic UAV swarms. The heterogeneity of data and complex communication environments in these swarms pose significant challenges for DIDS deployment, requiring systems to handle diverse device data and adapt to frequently shifting network topologies. This situation demands greater flexibility and adaptability from the system.

C. Federated Learning for Intrusion Detection

FL, as an emerging distributed machine learning technology, offers innovative solutions to the intrusion detection challenges in UAV swarms [35]. By enabling multiple devices to collaboratively train models while preserving data privacy, FL significantly enhances the efficiency and accuracy of detection systems [36]. This approach is particularly crucial in UAV swarms. It leverages data distributed across various devices for model training without requiring data centralization. Recent years have seen notable advancements in FL-based UAV intrusion detection techniques. He et al. [37] introduced a collaborative intrusion detection algorithm based on Generative Adversarial Networks (CGAN) and blockchain. This method enhances data generation with LSTM and combines distributed FL techniques to ensure data security, significantly improving intrusion detection accuracy in UAV networks. Hadi et al. [38] developed an autonomous collaborative intrusion detection system (UAV-CIDS) based on a Feedforward Convolutional Neural Network (FFCNN). This system achieves up to 98.23% accuracy in zero-day attack detection. It also includes a real-time event response system to enhance UAV network security. Wang et al. [39] proposed an asynchronous federated learning (AFL) framework based on Vertical Heterogeneous Networks (VHetNet). They combined the CA2C algorithm to optimize UAV selection. High Altitude Platform Stations (HAPS) are used as the central server for efficient intrusion detection model training. This method achieved high detection accuracy with low energy consumption and rapid response times. Zhang et al. [40] developed an evolving DIDS based on federated continual representation learning. The system employs supervised contrastive loss, global information-aware regularization loss, and variance-driven memory update strategies. This setup allows the system to continuously capture features of emerging attacks. The approach enhances the system's ability to detect novel attacks and enables DIDS to better adapt to evolving attack patterns and network environments. While FL has made significant strides in UAV network intrusion detection, its broader application still faces challenges such as dynamic network environments, heterogeneous data, and complex traffic patterns. The robustness of FL models in UAV swarms has not been fully validated, as frequent client joins and departures have been shown to cause instability in the system. Additionally, improving the model's adaptability to high-complexity network traffic data is essential for handling diverse traffic patterns effectively. In scenarios with Non-IID (non-independent and identically distributed) data, models must be highly adaptable to manage uneven data distribution across clients. Addressing these critical issues will enhance the effectiveness of FL in UAV swarms, improving its ability to counter increasingly complex security threats.

Algorithm 1 Federated Averaging for UAV Swarm

Input: Total number of UAVs N, number of selected UAVs K, fraction of UAVs participating in each round C, number of communication rounds T, learning rate η, local batch size b, number of local epochs E.
Server executes: initialize ω.
for each round $t = 0$ to $T-1$ **do**
  $K = \max(C \cdot N, 1)$
  $\mathcal{S}^t \leftarrow$ random set of K UAVs

  UAV Side (each UAV $k \in \mathcal{S}^t$):
  Split the local dataset D_k into batches of size b:
  $batches \leftarrow \text{split}(D_k, b)$
  for each local epoch $i = 1$ to E **do**
    for batch $b \in batches$ **do**
      $\omega_k^t \leftarrow \omega_k^t - \eta \nabla \ell(\omega_k^t; b)$
    end for
  end for
  return ω_k^t to server

  Server Side:
  for each UAV $k \in \mathcal{S}^t$ in parallel **do**
    $\omega_k^{t+1} \leftarrow \omega_k^t$
  end for
  $\omega^{t+1} \leftarrow \sum_{k=0}^{K-1} \frac{n_k}{n} \omega_k^{t+1}$
end for

III. Methodology

A. FedAvg in UAV Swarm Scenarios

Traditional FL algorithms aggregate models from multiple FL clients through a central FL server to generate a global model. For instance, FedAvg [41], as detailed in Algorithm 1, employs weighted averaging to aggregate client models.

TABLE I: List of Key Notations

Notation	Description
N	Total number of UAVs
C	Fraction of UAVs participating in each round
K	Number of selected UAVs
$\mathcal{S}^t$	Set of selected clients in round t
D_k	Local dataset of client k
T	Number of communication rounds
E	Number of local epochs
b	Local batch size
η	Global learning rate
η_ω	Learning rate of local models
η_θ	Learning rate of the global classifier
ω_k^t	Local model of client k in round t
φ_k^t	Local feature extractor of client k in round t
θ^t	Global classifier at round t
M_N	Affinity matrix
$\mathcal{N}_k^t$	Set of top n most similar clients to client k in round t
$\widetilde{\omega}_k^t$	Temporary local model of client k in round t
$\delta_{k,i}^t$	Weight in the affinity matrix between client k and client i in round t
$\widetilde{\mathcal{R}}_k^{t,s}$	Average feature representation for client k with label s at round t
$\mathcal{R}_k^{t,s}{}_{agg}$	Aggregated feature representations of client k in round t

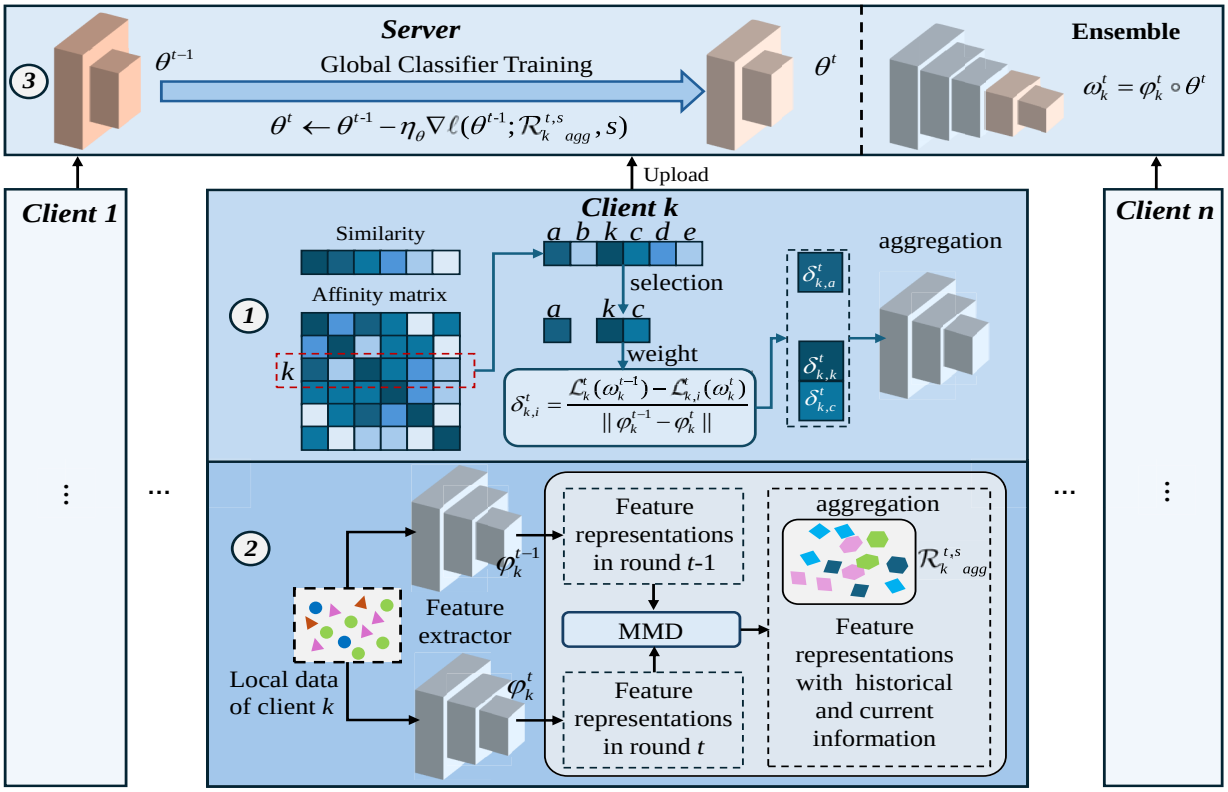


Fig. 1: FIDSUS System Framework

The server randomly selects a subset of clients in each round, referred to as UAVs in UAV-based applications. These selected UAVs, denoted by $\mathcal{S}^t$, train the global intrusion detection model ω on their local datasets D_k. The key notations used in this paper are summarized in Table I. The updated local models are then transmitted to the central server. At the server, these models are aggregated into a new global model using weighted averaging, as shown in the following expression:

$$\omega = \sum_{k=0}^{K-1} \frac{n_k}{n} \omega_k \tag{1}$$

FedAvg aims to solve the following optimization problem:

$$\min L(\omega) = \sum_{k=0}^{K-1} \frac{n_k}{n} L_k(\omega) \tag{2}$$

where n_k represents the number of samples in the local dataset of the k-th UAV, and n is the total number of samples across all UAVs. $L_k(\omega) = l(\omega; D_k)$ denotes the loss of the global intrusion detection model on the k-th UAV's local dataset D_k.

The performance of FedAvg relies on the assumption of homogeneous client data. However, in UAV swarm environments, data distributions are often dynamic and highly heterogeneous. Additionally, network traffic data for intrusion detection features complex characteristics. In such scenarios, the performance of FedAvg frequently falls short. To address this challenge, we propose FIDSUS, a federated intrusion detection framework designed for UAV swarms.

B. FIDSUS Framework Overview

The key steps of FIDSUS are summarized as follows:

Feature Extraction: The first step involves decomposing the model into a feature extractor and a classifier. The purpose of this step is to capture important features from each client's local dataset. To improve the quality of feature extraction, we construct an affinity matrix, which measures the similarity between local clients based on their data. This matrix serves as a tool to guide the aggregation of local feature extractors, enhancing detection accuracy by improving knowledge sharing across heterogeneous clients.

Feature Aggregation: During each iteration, the feature extractor processes the client's local dataset and collects feature representations from both the previous and current rounds. To measure and quantify any shifts in feature distributions across iterations, we employ MMD [26]. This allows us to detect discrepancies between feature representations from different rounds and apply these discrepancies as weights in the aggregation process. The rationale behind this approach is to ensure that the feature aggregation prioritizes the most relevant and stable features over time.

Global Classification: After the feature representations are updated, they are transmitted to the server, which integrates them to train the global classifier. The objective here is to leverage the feature representations from all participating clients to build a robust global classifier. By doing so, we enhance the generalization capability of the model, enabling it to perform well across a variety of dynamic conditions. Since MMD aggregates historical feature representations, the global

classifier also incorporates this historical data to mitigate the problem of forgetting previously learned knowledge. By retaining information from past rounds, we ensure that the model maintains performance on patterns learned earlier, contributing to the classifier's stability and improving its generalization ability over time.

To effectively detect highly heterogeneous data, FIDSUS utilizes a personalized federated learning (PFL) strategy. PFL allows each participant to adjust the local model according to their own needs and characteristics. By deploying personalized local models, FIDSUS enables clients to better adapt to their unique data distributions, surpassing the limitations of a single global model. The proposed FIDSUS framework is depicted in Fig. 1. FIDSUS aims to minimize the total loss across all client models on their local datasets, formulated as:

$$\min \sum_{k=0}^{N-1} \mathcal{L}_k \left(\omega_k\right) \tag{3}$$

where $\mathcal{L}_k \left(\omega_k\right)$ represents the loss of the k-th client on its local dataset, and ω_k denotes the local model parameters of the k-th client. To achieve precise feature extraction and classification, we decouple the model ω into a feature extractor φ and a classifier θ, expressed as:

$$\omega = \varphi \circ \theta \tag{4}$$

where φ is the feature extractor, θ is the classifier, and $\circ$ denotes model concatenation. The system aggregates φ to enhance feature extraction capabilities and trains the global classifier θ to improve the system's generalization performance. The detailed steps are shown in Algorithm 2.

Before the iterations begin, we randomly initialize the heterogeneous local models $\left[\omega_0^0, \ldots, \omega_{N-1}^0\right]$ and the global classifier θ^0, while also initializing M_N as a diagonal matrix. In the t-th round of iteration, K clients are randomly selected to participate in training, denoted as $\mathcal{S}^t$. Each selected client $k \in S^t$ selects the top n most similar clients, denoted by $\mathcal{N}_k^t$, based on the weights from the affinity matrix M_N. Subsequently, client k collects feature extractors $< \varphi_{k,1}^t, \ldots, \varphi_{k,n}^t >$ from clients in $\mathcal{N}_k^t$. The server receives these feature extractor parameters and distributes them to the selected client k. The client combines its local classifier with the sampled feature extractors to form a complete model with different feature extractors:

$$\begin{aligned} \omega_{k,1}^t &= \varphi_{k,1}^t \circ \theta^{t-1} \\ &\vdots \\ \omega_{k,n}^t &= \varphi_{k,n}^t \circ \theta^{t-1} \end{aligned} \tag{5}$$

At this stage, each sampled client has $n+1$ models, including its local model $\omega_k^{t-1} = \varphi_k^{t-1} \circ \theta^{t-1}$. The client evaluates these models to update the weights in the affinity matrix M_N. The updated local feature extractor φ_k^t is then computed as follows:

$$\varphi_k^t = \varphi_k^{t-1} + \sum_{i \in \mathcal{N}_k^t} \hat{\delta}_{k,i} \left(\varphi_{k,i}^t - \varphi_k^{t-1}\right) \tag{6}$$

Algorithm 2 FIDSUS

Input: Total number of UAVs N, number of selected UAVs K, number of communication rounds T, learning rate of local models η_ω, learning rate of global classifier η_θ.
Server executes: randomly initialize the heterogeneous local models $\left[\omega_0^0, \ldots, \omega_{N-1}^0\right]$ and global classifier θ^0. Initialize affinity matrix $M_N = \mathrm{diag}(1, 1, \ldots, 1)$

for each round $t = 0$ to $T-1$ **do**
$\mathcal{S}^t \leftarrow$ Randomly select $K \leq N$ UAVs to join FL.

UAV Side (each UAV $k \in \mathcal{S}^t$):
Choose top n similar UAVs based on M_N.
Receive feature extractors $< \varphi_{k,1}^t, \ldots, \varphi_{k,n}^t >$ and global classifier θ^{t-1} from the server.
Concatenate each received feature extractors with the global classifier by Eq. (5).
Update weight vector of M_N using **Algorithm 3**.
Update local feature extractor φ_k^t by Eq. (6).
Generate a temporary local model $\widetilde{\omega}_k^t = \varphi_k^t \circ \theta^{t-1}$
Update local model ω_k^t using gradient descent by Eq. (7).
Upload the local class representation to the server according to **Algorithm 4**.

Server Side:
Update M_N using **Algorithm 3**.
Train global classifier θ^t by **Algorithm 4**.
Broadcast θ^t to UAVs for the next round.
end for
Return Personalized models: $\left[\omega_0^{T-1}, \omega_1^{T-1} \ldots, \omega_{N-1}^{T-1}\right]$.

where $\widehat{\delta}_{k,i}$ represents the normalized weight, which is used to compute the weighted sum of the differences between $\varphi_{k,i}^t$ and φ_k^{t-1}. The updated local feature extractor φ_k^t is then combined with the previous round's global classifier θ_k^{t-1} to form a temporary local model $\widetilde{\omega}_k^t$. The local model ω_k^t is subsequently updated through gradient descent on the loss function as follows:

$$\omega_k^t \leftarrow \widetilde{\omega}_k^t - \eta_\omega \nabla \ell \left(\widetilde{\omega}_k^t; D_k\right) \tag{7}$$

where D_k denotes the local dataset held by client k, and η_ω is the local learning rate. After updating the local feature extractor, $\widetilde{\mathcal{R}}^{t-1,s}$ and $\widetilde{\mathcal{R}}^{t,s}$ are used to form a new aggregated feature representation $\widetilde{\mathcal{R}}^{t,s}{}_{agg}$. The server collects all new feature representations $\widetilde{\mathcal{R}}^{t,s}{}_{agg}$ to train the global classifier θ^t and distribute it to clients.

C. Update of Local Feature Extractors

The algorithm for local feature extractor updates, including the affinity matrix update, is shown in Algorithm 3. Clients with similar data distributions can optimize local model performance through knowledge sharing. To quantify the similarity between clients, we establish an N-dimensional client affinity matrix M_N. For the models $\omega_{k,1}^t = \varphi_{k,1}^t \circ \theta^{t-1}, \ldots, \omega_{k,n}^t = \varphi_{k,n}^t \circ \theta^{t-1}$ and $\omega_k^{t-1} = \varphi_k^{t-1} \circ \theta^{t-1}$ obtained in Eq. (5), we

evaluate these $n+1$ models on the local model to update the weights in the affinity matrix M_N. First, we calculate the loss function value of the local model ω_k^{t-1}, denoted as $\mathcal{L}_k^t\left(\omega_k^{t-1}\right)$. Next, we compute the loss function values $\mathcal{L}_{k,i}^t\left(\omega_{k,i}^t\right)$ for each $\omega_{k,i}^t$. The weight update formula is then applied as follows:

$$\delta_{k,i}^t = \frac{\mathcal{L}_k^t\left(\omega_k^{t-1}\right) - \mathcal{L}_{k,i}^t\left(\omega_{k,i}^t\right)}{\left\|\varphi_k^{t-1} - \varphi_{k,i}^t\right\|}, (i \in \mathcal{N}_k^t) \tag{8}$$

The new weight is defined based on the difference between the loss function values $\mathcal{L}_k^t\left(\omega_k^{t-1}\right)$ and $\mathcal{L}_{k,n}^t\left(\omega_{k,i}^t\right)$, normalized by the L_2 norm of the difference between the feature extractor parameters φ_k^{t-1} and $\varphi_{k,i}^t$. A larger difference indicates that the local model ω_k^{t-1} performs worse on the local validation set compared to the received model $\omega_{k,i}^t$. This results in a higher weight for φ_k^{t-1}.

To ensure numerical stability and avoid situations where the denominator approaches zero due to the cancellation of positive and negative terms, the result of Eq. (8) is processed with $\delta_{k,i}^t \leftarrow \max(0, \delta_{k,i}^t)$, followed by normalization of the positive values. This ensures that the computation remains well-defined and robust, particularly in cases where the differences in parameters are small.

When the difference between φ_k^{t-1} and $\varphi_{k,i}^t$ is small, the shared knowledge can provide a greater benefit, leading to a higher weight for $\varphi_{k,i}^t$. Conversely, when the difference is large, the weight is relatively smaller, reflecting the limited potential improvement for the local model. Subsequently, normalization is applied to prevent instability caused by excessively large or small weights:

$$\widehat{\delta}_{k,i}^t = \frac{\delta_{k,i}^t}{\sum_{i \in \mathcal{N}_k^t} \delta_{k,i}^t} \tag{9}$$

The updated feature extractor φ_k^t is obtained using the normalized weights as follows:

$$\varphi_k^t = \varphi_k^{t-1} + \sum_{i \in \mathcal{N}_k^t} \hat{\delta}_{k,i}\left(\varphi_{k,i}^t - \varphi_k^{t-1}\right) \tag{10}$$

After updating the weights $\left[\hat{\delta}_{k,1}^t, \ldots, \hat{\delta}_{k,n}^t\right]$ for the n clients similar to client k, the weight vector δ_k^t is updated as:

$$\delta_k^t \leftarrow \left[\hat{\delta}_{k,1}^t, \ldots, \hat{\delta}_{k,n}^t\right] \tag{11}$$

The server then collects the updated weight vectors $< \delta_1^t, \ldots, \delta_k^t >$ from all participating clients and updates the affinity matrix M_N. This updated matrix facilitates the selection of similar clients for the subsequent iteration. By incorporating model updates from other clients, the local feature extractors can learn relevant information more efficiently. Weighting the extractors based on their relative contributions accelerates model convergence and enhances performance.

D. Update of Global Classifier

To enhance the system's generalization and adaptability, we train the global classifier using feature representations. For client k, the updated feature extractors φ_k^t and the previous feature extractor φ_k^{t-1} are used to extract new and old feature representations, respectively. Feature representations $\mathcal{R}$ are numerical vectors that capture the essential characteristics of data segments.

To align new and old features, we employ MMD to measure distributional differences. MMD is a statistical method that quantifies the discrepancy between two probability distributions, reflecting the differences between new and old feature representations. In FL, where data distributions can be highly heterogeneous and complex, MMD effectively captures these distributional differences [26]. The MMD formula is:

$$\begin{aligned} MMD(P,Q) = &\frac{1}{n_x^2}\sum_{p=1}^{n_x}\sum_{q=1}^{n_x} k\left(X_p, X_q\right) + \\ &\frac{1}{n_y^2}\sum_{p=1}^{n_y}\sum_{q=1}^{n_y} k\left(Y_p, Y_q\right) + \frac{2}{n_x n_y}\sum_{p=1}^{n_x}\sum_{q=1}^{n_y} k\left(X_p, Y_q\right) \end{aligned} \tag{12}$$

where P and Q represent the distributions of X and Y, n_x and n_y denote the number of samples in X and Y, respectively, and $k(\cdot,\cdot)$ is the kernel function that measures the similarity between samples. This formula captures the differences between the internal self-similarity of X and Y as well as their cross-similarity.

Since feature representations are typically multi-dimensional, directly computing MMD significantly increases computational complexity and time cost. The time complexity for direct computation of MMD is $O\left(n_x^2 + n_y^2 + n_x n_y\right)$. To reduce this complexity, we use an average feature representation approach. For the category s in the local dataset D_k of client k, the average feature representations are

---

Algorithm 3 Update of Local Feature Extractors

Input: Number of UAVs selected based on the affinity matrix n, local model from the previous round ω_k^{t-1}, feature extractors of other selected models $< \varphi_{k,1}^t, \ldots, \varphi_{k,n}^t >$, global classifier received from server θ^{t-1}.

UAV Side (each UAV $k \in \mathcal{S}^t$):
Calculate affinity weights $\hat{\delta}_{k,i}^t$ by Eq. (8)
Normalize weights $\hat{\delta}_{k,i}^t$ by Eq. (9)
Update local feature extractor $\hat{\delta}_{k,i}^t$ by Eq. (10)
Update weight vector δ_k^t :
$\delta_k^t \leftarrow \left[\hat{\delta}_{k,1}^t, \ldots, \hat{\delta}_{k,n}^t\right]$
Upload $\hat{\delta}_{k,i}^t$ to server.
Update the local model

Server Side:
Receive $< \delta_1^t, \ldots, \delta_k^t >$
Update the affinity matrix M_N:
$M_N \leftarrow \begin{bmatrix} \delta_1^t \\ \vdots \\ \delta_k^t \end{bmatrix}$

---

computed as follows:

$$\widetilde{\mathcal{R}}_k^{t-1,s} = \frac{1}{|D_k^s|} \sum_{i \in D_k^s} \mathcal{F}_k^{t-1}\left(\varphi_k^{t-1}; x_i\right) \tag{13}$$

$$\widetilde{\mathcal{R}}_k^{t,s} = \frac{1}{|D_k^s|} \sum_{i \in D_k^s} \mathcal{F}_k^{t}\left(\varphi_k^{t}; x_i\right) \tag{14}$$

where $\mathcal{F}_k^t\left(\varphi_k^t; x_i\right)$ represents the feature extraction using φ_k^t on a subset of samples x_i from the local dataset D_k^s. This process yields average feature representations $\widetilde{\mathcal{R}}_k^{t-1,s}$ and $\widetilde{\mathcal{R}}_k^{t,s}$ for categories s in rounds $t-1$ and t, respectively. With average feature representations, Eq. (12) simplifies to:

$$MMD\left(\widetilde{\mathcal{R}}_k^{t,s}, \widetilde{\mathcal{R}}_k^{t-1,s}\right) = 2 - k\left(\widetilde{\mathcal{R}}_k^{t,s}, \widetilde{\mathcal{R}}_k^{t-1,s}\right) \tag{15}$$

This reduction in complexity simplifies the computation to $O\left(n_x + n_y\right)$, significantly lowering the cost compared to direct MMD computation.

Algorithm 4 Update of Global Classifier

Input: Total number of UAVs N, learning rate of global classifier η_θ, feature extractor from the previous round φ_k^{t-1}, feature extractor aggregated in Algorithm 3 φ_k^t

UAV Side (each UAV $k \in \mathcal{S}^t$):
Calculate representations $\mathcal{R}_{k,i}^{t-1}$ and $\mathcal{R}_{k,i}^{t}$ for each sample $i \in D_k$.
Calculate average representations for each class using Eq. (13) and Eq. (14).
Calculate MMD between $\tilde{\mathcal{R}}_k^{t-1,s}$ and $\tilde{\mathcal{R}}_k^{t,s}$ using Eq. (15).
Update aggregation weights W.
Calculate aggregated representation by Eq. (16).
Upload $\mathcal{R}_k^{t,s}{}_{agg}$ and class labels s to the server.
Receive aggregated representations $\mathcal{R}_k^{t,s}{}_{agg}$ and label s from clients. Train the global classifier:

Server Side:
Receive aggregated representations $\mathcal{R}_k^{t,s}{}_{agg}$ and label s from clients.
Train the global classifier:
$\theta^t \leftarrow \theta^{t-1} - \eta_\theta \nabla \ell\left(\theta^{t-1}; \mathcal{R}_k^{t,s}{}_{agg}, s\right)$
Broadcast the updated global classifier θ^t

The normalized difference values are denoted as weights W. These weights reflect the relative importance of each local representation. They help in capturing the underlying data distribution. These weights balance the contributions of historical feature representations $\widetilde{\mathcal{R}}_k^{t-1,s}$ and current feature representations $\widetilde{\mathcal{R}}_k^{t,s}$. We use a weighted combination to aggregate $\widetilde{\mathcal{R}}_k^{t-1,s}$ and $\widetilde{\mathcal{R}}_k^{t,s}$:

$$\mathcal{R}_k^{t,s}{}_{agg} = W \cdot \widetilde{\mathcal{R}}_k^{t,s} + (1 - W)\widetilde{\mathcal{R}}_k^{t-1,s} \tag{16}$$

where $\mathcal{R}_k^{t,s}{}_{agg}$ represents the aggregated average representation for each local class in round t, used for training the global classifier. This process ensures that both current feature representations $\widetilde{\mathcal{R}}_k^{t,s}$ and historical representations $\widetilde{\mathcal{R}}_k^{t-1,s}$ are integrated to enhance the model's generalization ability and adaptability. By uploading feature representations $\mathcal{R}_k^{t,s}{}_{agg}$, FIDSUS avoids the inefficiency of directly transmitting entire models which is beneficial in resource-constrained environments. During aggregation, UAV clients with feature distributions significantly deviating from historical distributions are assigned lower weights. This approach prevents the unified feature representation from being overly influenced by outliers or domain biases. Aggregating both old and new feature representations preserves historical information, mitigating the loss of useful features and patterns. This method helps maintain a memory of previously learned information, allowing better utilization of past experiences and knowledge.

On the server side, the aggregated feature representations $\mathcal{R}_k^{t,s}{}_{agg}$ from all participating clients in the current round t are used to train the global classifier. Parameter updates are performed by calculating the gradient of the loss function ℓ with respect to the global classifier θ^{t-1}. The update formula is:

$$\theta^t \leftarrow \theta^{t-1} - \eta_\theta \nabla \ell\left(\theta^{t-1}; \mathcal{R}_k^{t,s}{}_{agg}, s\right) \tag{17}$$

where η_θ is the learning rate for the global classifier, and $\nabla \ell\left(\theta^{t-1}; \mathcal{R}_k^{t,s}{}_{agg}, s\right)$ represents the gradient of the loss function ℓ with respect to the global classifier θ^{t-1}, based on the aggregated feature representations $\mathcal{R}_k^{t,s}{}_{agg}$ and category s. The updated global classifier θ^t is then distributed to the selected clients in the next round.

The global classifier integrates information from multiple models, enhancing prediction accuracy. By leveraging diverse perspectives, this approach mitigates individual model biases, improves robustness, and reduces overfitting, ultimately leading to better generalization.

IV. Experiment and analysis

A. Experimental Configuration

To evaluate the effectiveness of the proposed FIDSUS framework, we conducted comparative experiments using two network intrusion detection datasets (NSL-KDD [42] and UNSW-NB15 [43]) and seven advanced methods (FedAvg [41], FedProx [44], MOON [45], FedAvgDBE [46], FedProto [47], GPFL [48], and FedGH [49]) across various scenarios. For all FL strategies, we employ the same 1D CNN model for intrusion detection. The model architecture comprises two convolutional layers, one global max pooling layer, one dropout layer, and one fully connected layer. All methods employed consistent hyperparameter settings. The specific parameters are as follows: both the local learning rate and the learning rate of the global classifier on the server are set to 0.01, with a batch size of 64. The join ratio is set to 1. The number of communication rounds is set to 100, and the local epoch is set to 1. In the FedProx and MOON methods, the value of μ is set to 0.01, while in MOON, τ is set to 1. In GPFL and FedProto, λ is set to 0.99. In FedAvgDBE, the momentum is set to 0.1, and the kl_weight is set to 0 by default. These parameter settings were determined based on preliminary experiments. All experiments were conducted on an RTX 4090 GPU using the PyTorch 1.8 framework with Torch version 2.0.1.

Comparison Strategies:

1) **FedAvg** aims to update the global model by averaging local model parameters from different clients. After

each communication round, the server collects the local model parameters from all clients, computes the average, and then uses this average as the update for the global model.

2) **FedProx** builds upon FedAvg by introducing a regularization term to balance the accuracy of the global model and the deviation of the client models, thereby enhancing the model's generalization ability. It addresses performance issues of FedAvg in heterogeneous data situations.
3) **MOON** integrates global model and client model features to address performance issues on non-IID data. It uses a mixed optimization algorithm to simultaneously update parameters of the global model and client models and employs feature selection methods to weight and fuse features from different models.
4) **FedAvgDBE** enhances bidirectional knowledge transfer between server and clients. This approach eliminates domain bias in feature extractors and addresses catastrophic forgetting during local training.
5) **FedProto** transmits abstract class prototypes between clients and the server. Local prototypes from different clients are collected and aggregated to form a global prototype. This global prototype then guides the training of local models, ensuring continuous alignment between local and global prototypes.
6) **GPFL** extracts global features using global model parameters at the clients and local features from private data. These features are then fused to obtain a combined feature, which is used for model training to produce the global model.
7) **FedGH** utilizes heterogeneous feature extractors to obtain feature representations and trains a shared generalized global prediction head. This head learns from different clients and then transfers the acquired global knowledge back to the clients to replace each client's local prediction head.

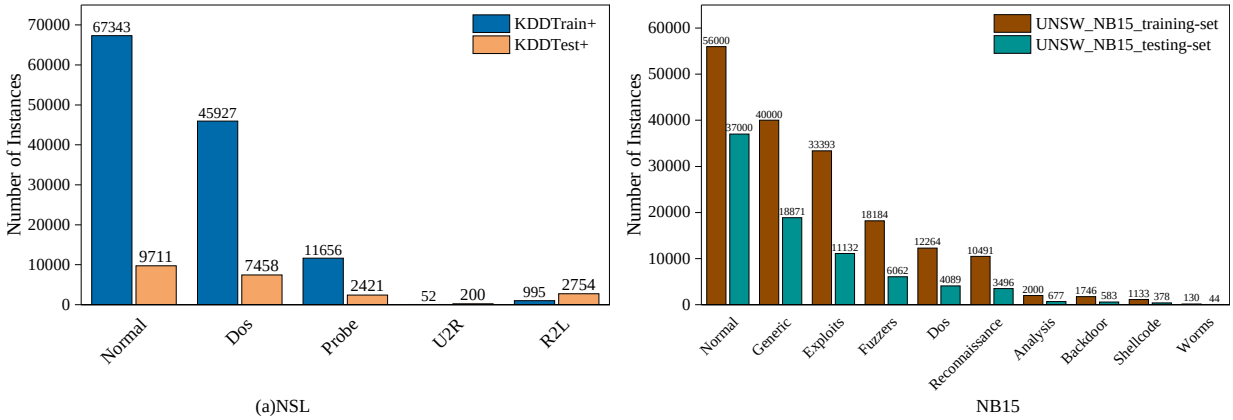
B. Experimental Datasets for Intrusion Detection

To evaluate the performance of the proposed FIDSUS framework, we used two popular network intrusion detection datasets: UNSW-NB15 and NSL-KDD. The UNSW-NB15 dataset comprises records with 49 features, covering 10 different types of attacks. In contrast, each record in the NSL-KDD dataset contains 43 features. The NSL-KDD dataset's training set includes 22 attack types, while its test set expands to 39 attack types. For the 39 attack types in the NSL-KDD dataset, we reclassified them based on their nature using the mapping scheme outlined in Table II. Since the UNSW-NB15 dataset is derived from real network traffic and specifically designed to address the complex attack behaviors in modern network environments, it is better suited to represent the complexities of contemporary cyberattacks.

Fig. 2 shows the distribution of data quantities for different attack types in these two datasets. After classifying the attack types, as shown in Fig. 2(a), the NSL-KDD dataset is divided into five categories: Normal, Dos, Probe, U2R, and R2L. As illustrated in Fig. 2(b), the UNSW-NB15 dataset consists of ten categories: Normal, Generic, Exploits, Fuzzers, Dos, Reconnaissance, Analysis, Backdoor, Shellcode, and Worms.

TABLE II: Mapping of Attack Classes in NSL-KDD

Class	Description	Attack Category
Normal	Normal network traffic	normal
Dos	Dos attacks make the target inaccessible by sending a large amount of traffic or information to the target server.	apache2, back, mailbomb, processtable, snmpgetattack, teardrop, smurf, land, neptune, pod, udpstorm
Probe	Collect network information	nmap, ipsweep, portsweep, satan, mscan, saint, worm
U2R	Obtaining root privileges through illegal means	ps, buffer_overflow, perl, rootkit, loadmodule, xterm, sqlattack, http-tunnel
R2L	Remote user attacks exploit security vulnerabilities and perform illegal operations by remotely logging into the computer.	ftp_write, guess_passwd, snmpguess, imap, spy, warezclient, warezmaster, multihop, phf, named, sendmail, xlock, xsnoop



(a)NSL NB15

Fig. 2: The Distribution of Attack Types (a)NSL-KDD (b)UNSW-NB15

C. Data Preprocessing

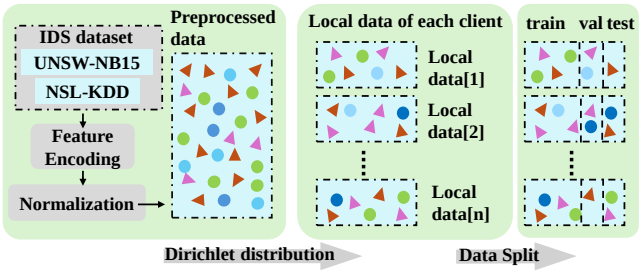


Fig. 3: Data Preprocessing

First, we encode categorical features (such as protocol, flag, etc.) by mapping them into a numerical space. For encoding, one-hot encoding is applied for categorical features with no intrinsic order, where each category is represented as a separate binary feature. Label encoding is applied to categorical features with an ordinal relationship where each category is mapped to a unique integer value. This ensures that categorical features are transformed into a format suitable for FL models, allowing them to process the data effectively. Next, features with significant scale differences are normalized. Min-Max scaling is applied, rescaling each feature to a range between 0 and 1, ensuring all features are on a comparable scale. This step mitigates the impact of large disparities in

Then, we use the Dirichlet distribution to partition the local datasets and simulate heterogeneous Non-IID scenarios. This distribution helps create imbalanced data splits, which more closely mimic real-world federated learning environments where clients may have data of varying quantities and distributions. The Dirichlet distribution is defined over a set of multinomial random variables, with its probability density function given by:

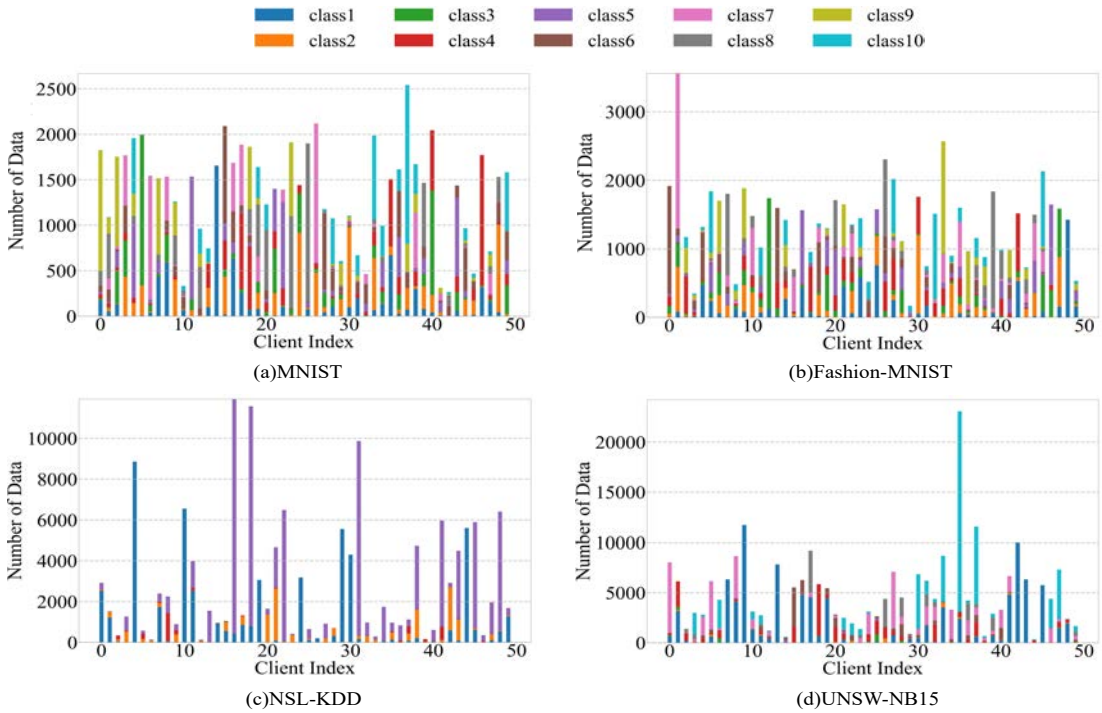
$$\mathrm{Dir}(p \mid \alpha) = \frac{1}{B(\alpha)} \prod_{i=1}^{K} p_i^{\alpha_i - 1} \tag{18}$$

where $p = (p_1, p_2, \ldots, p_K)$ is a K-dimensional random variable vector, with $0 \leq p_i \leq 1$ and $\sum_{i=1}^{K} p_i = 1$. The parameter vector $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_K)$ controls the shape of the probability density function across the dimensions, reflecting preferences or weights in each dimension. $B(\alpha)$ denotes the multivariate Beta function, which ensures that the probability density function integrates to 1, reflecting the normalization property of the Dirichlet distribution. The parameter α is set to 0.3, and this value is applied in all subsequent experiments.

After applying the Dirichlet distribution to partition the dataset, each client receives a local dataset. This dataset is further divided into three subsets with a ratio of 4:1:1. The 4:1:1 ratio is chosen to strike a balance between having enough data for model training and maintaining an unbiased evaluation setup.

D. Experimental Results Analysis with Static UAV Clients

1) Comparison of Datasets: Fig. 4 compares the data distribution across the MNIST [50], FashionMNIST [51], NSL-



(a)MNIST (b)Fashion-MNIST

(c)NSL-KDD (d)UNSW-NB15

Fig. 4: Client Subsets under Dirichlet Distribution

(NC) is set to 50 and the Dirichlet distribution parameter held constant. While MNIST and FashionMNIST exhibit relatively balanced class distributions, the NSL-KDD and UNSW-NB15 datasets show significant imbalances, resulting in greater data heterogeneity. This heterogeneity presents challenges for FL systems, particularly in managing variations in the number of classes and samples per client. Comparing Figs. 4(a) and 4(b) with Figs. 4(c) and 4(d) reveals that the inherent imbalance in the original datasets exacerbates the heterogeneity, especially in network traffic data. This imbalance introduces additional difficulties in processing and training within FL frameworks.

Fig. 5 illustrates the performance differences of various FL strategies across the MNIST, FashionMNIST, NSL-KDD, and UNSW-NB15 datasets. The results show relatively stable performance curves for the MNIST and FashionMNIST image datasets, where the simpler and lower-dimensional nature of the data leads to smoother convergence. In contrast, the intrusion detection datasets exhibit greater fluctuations during training, primarily due to the high dimensionality and complexity of network traffic data. These characteristics, combined with the inherent class imbalance in the intrusion detection datasets, contribute to the instability in convergence. As observed in Figs. 4(a) through 4(d), more imbalanced datasets lead to increasingly skewed client subsets, even when a consistent Dirichlet distribution is applied. Among the various FL strategies, FedGH is the most affected by this imbalance, demonstrating strong performance on image tasks but struggling with stability in intrusion detection tasks. This highlights the challenges posed by the inherent complexity and imbalance of network traffic data, which require the development of more robust strategies to ensure effective learning and consistent system performance across different domains.

2) Performance Evaluation on the NSL-KDD Dataset: Fig. 6 illustrates the performance of various federated strategies on the NSL-KDD dataset. This dataset, characterized by its relatively low complexity, is well-suited for simulating low-complexity traffic scenarios involving UAVs. The proposed FIDSUS strategy consistently outperforms others across all scenarios, achieving the highest average test accuracy with a stable upward trend. FIDSUS evaluates the affinity matrix to assess the similarity between UAV clients, enabling knowledge sharing and enhanced aggregation among similar UAVs. This allows the model to quickly adapt to the local data of each UAV client, effectively addressing the challenges posed by high heterogeneity in recognition tasks. FIDSUS also leverages feature representations from both before and after client updates. By aggregating these features, it incorporates both historical and current information, thereby mitigating the issue of forgetting in highly heterogeneous environments. The final step involves training a global classifier with these enriched feature representations. This process enhances the model's overall generalization and ensures stable average test accuracy across different scales. In contrast, other strategies such as FedAvg, FedProx, MOON, FedProto, FedAvgDBE, and GPFL

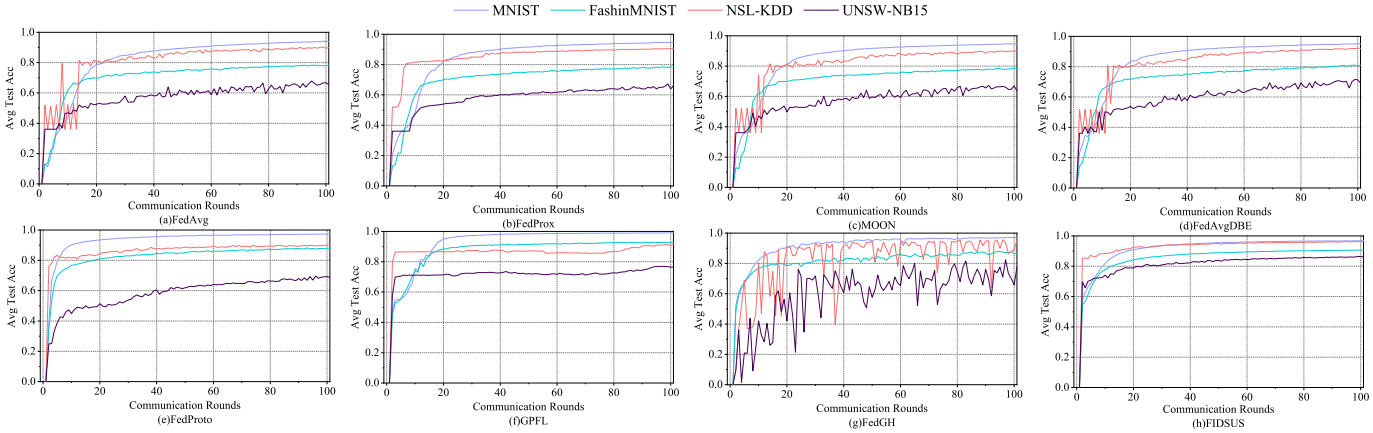


Fig. 5: Performance Comparison Across Different Datasets

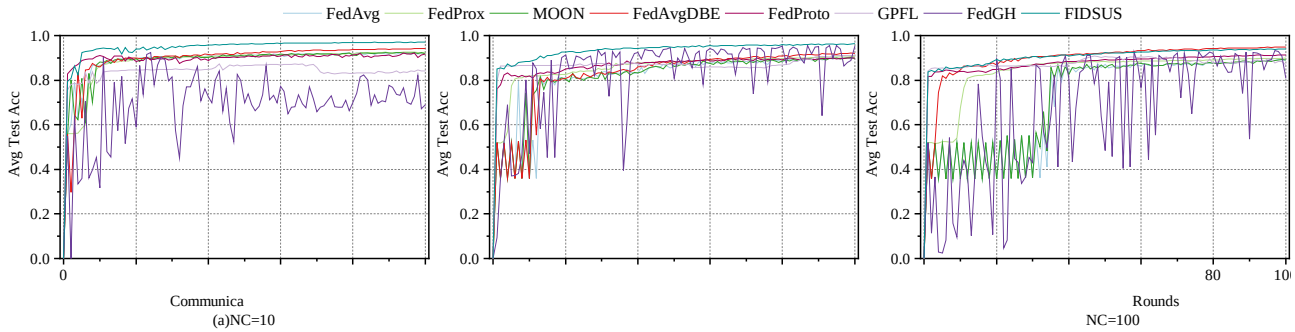


Fig. 6: Performance Comparison on the NSL-KDD Dataset

exhibit varying levels of performance. FedAvg updates the global model through parameter averaging, showing stable performance and excelling on low-complexity datasets.

FedProx introduces a regularization term to FedAvg, which initially causes slight fluctuations in accuracy due to adjustments in weight balance between client models and the global model. This approach eventually stabilizes and achieves high accuracy as well. MOON, which integrates features from both global and client models, experiences significant initial oscillations but ultimately converges to high accuracy. FedAvgDBE improves the bidirectional knowledge transfer between the server and UAVs, reducing domain bias and maintaining high accuracy. FedProto reduces client differences by sharing abstract class prototypes. It shows limited improvement on low-complexity datasets but demonstrates stability as the number of clients increases. GPFL shows high initial average test accuracy, but the curve flattens later. This indicates that while GPFL excels at quickly achieving high detection rates on low-complexity data, its performance reaches a ceiling with continued communication rounds. FedGH experiences oscillatory growth throughout, struggling with integration challenges under highly heterogeneous data, resulting in suboptimal performance.

3) Performance Evaluation on the UNSW-NB15 Dataset: Fig. 7 illustrates the performance of various strategies on the UNSW-NB15 dataset. Our proposed FIDSUS strategy consistently achieves the highest detection accuracy across all UAV scales with minimal fluctuations. FIDSUS effectively addresses challenges posed by higher data complexity through dynamic evaluation of similarity between UAVs and knowledge sharing based on an affinity matrix. By aggregating feature representations that incorporate both historical and current data, FIDSUS mitigates the issue of forgetting. This approach helps maintain stable accuracy across different scales even in complex scenarios. GPFL initially excels due to feature integration, but its performance plateaus in later rounds due to challenges in feature fusion. This limits its ability to handle complex data effectively. FedGH's aggregation strategy faces challenges on the UNSW-NB15 dataset. Although it reaches 80% accuracy with 10 UAVs, it experiences significant fluctuations overall. In high-complexity data scenarios, FedGH's generalized global prediction head struggles with feature integration, resulting in considerable performance volatility. FedProx demonstrates stable performance with 10 UAVs, achieving accuracy similar to MOON at 73%. However, as the number of UAVs increases to 50, FedProx's performance declines, though it still outperforms most other strategies. FedProx's regularization term balances biases between global and local models, but it struggles with the challenges posed by heterogeneity. MOON attempts to address Non-IID data by integrating global and local model features. However, on the UNSW-NB15 dataset, its feature fusion strategy may lead to fluctuations and performance instability in complex data scenarios. FedProto experiences a significant drop in performance, falling below 50% accuracy when the number of UAVs reaches 100. While it reduces inter-client differences by transmitting abstract class prototypes, the difficulty of aggregating these prototypes in large-scale UAV scenarios leads to a sharp decline in accuracy. Similarly, both FedAvg and FedAvgDBE show declines in performance when the number of UAVs reaches 100, with average test accuracy stabilizing around 60% after 100 iterations. FedAvg's parameter averaging design maintains some stability but struggles to achieve optimal results in complex scenarios. FedAvgDBE aims to address domain bias through bidirectional knowledge transfer. However, it fails to fully integrate client data features with the global model's knowledge in complex settings, leading to suboptimal performance in certain domains. In contrast, FIDSUS excels across different UAV scales, particularly with high-complexity datasets like UNSW-NB15. Its efficient aggregation strategy and superior adaptability highlight its effectiveness in handling challenging scenarios. While other strategies have their strengths and weaknesses, FIDSUS's stability and accuracy become more pronounced in complex data scenarios.

4) Performance Comparison in Low Complexity Small-Scale Scenario: In small-scale UAV client scenarios, we assessed the performance of different FL strategies with CAR values of 1.0 and 0.5. For medium-scale UAV client scenarios, we evaluated performance with CAR values of 1.0, 0.9, 0.7, and 0.5. The experimental results demonstrate the performance of various FL strategies in dynamic scenarios.

Table III presents the highest accuracy achieved by each method across six different scenarios within 100 communication rounds. Our proposed FIDSUS achieved the highest accuracy in five of these scenarios, demonstrating its robust performance in handling varying data scales and complexities.

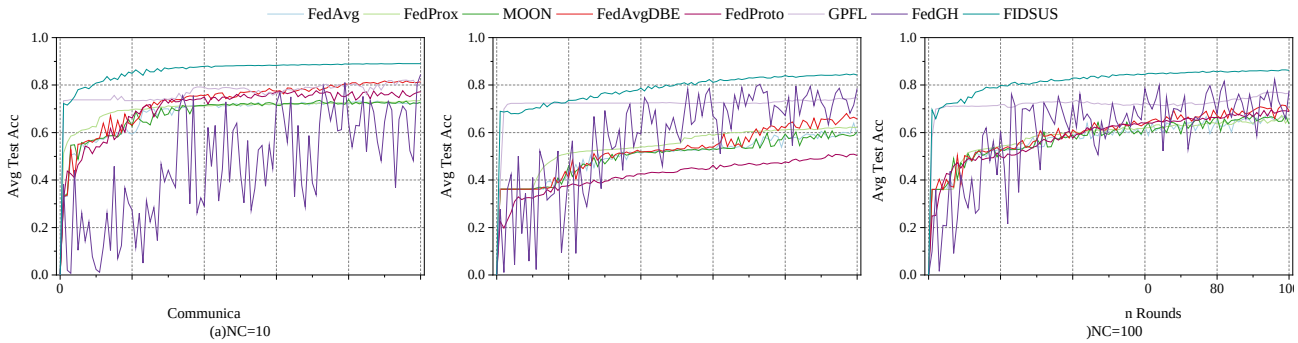


Fig. 7: Performance Comparison on the UNSW-NB15 Dataset

FIDSUS excels in processing highly heterogeneous data by dynamically calculating similarity between UAVs and leveraging an affinity matrix for knowledge sharing and aggregation. Additionally, FIDSUS improves model generalization and stability by integrating historical and current data features, enabling it to perform well in both low- and high-complexity scenarios.

TABLE III: Best Accuracy in 100 Communication Rounds

Method	NSL-KDD			UNSW-NB15		
	NC=10	NC=50	NC=100	NC=10	NC=50	NC=100
FedAvg	0.9255	0.9100	0.8922	0.7333	0.6781	0.6391
FedProx	0.9274	0.9070	0.8985	0.7359	0.6722	0.6230
MOON	0.9241	0.9012	0.8929	0.7387	0.6727	0.6072
FedAvgDBE	0.9419	0.9219	0.9490	0.8175	0.7161	0.6797
FedProto	0.9215	0.9001	0.9140	0.7764	0.6963	0.5085
GPFL	0.8737	0.9115	0.9074	0.8216	0.7696	0.7483
FedGH	0.9247	0.9562	0.9339	0.8447	0.8238	0.8037
FIDSUS	**0.9712**	**0.9634**	**0.9398**	**0.8910**	**0.8645**	**0.8467**

E. Experimental Results Analysis with Dynamic UAV Clients

To evaluate the impact of mobile scenarios on different FL strategies, we introduced the Client Activity Rate (CAR). During each aggregation round, only active clients update the parameters. This simulates the loss and rejoining of clients in mobile environments. By evaluating the performance of various methods under different CAR parameters, we compared their accuracy and robustness in these mobile scenarios.

Fig. 8 illustrates the performance comparison of various methods in a low-complexity scenario with 10 clients, where the CAR is set to 1.0 and 0.5. Each subplot represents a different FL strategy. As shown in Fig. 8, FedAvg, FedProx, MOON, and FedAvgDBE all experience notable performance drops during specific rounds when clients are lost. Their performance eventually recovers to normal levels. Such fluctuations can pose significant issues in real-world scenarios. These strategies are heavily reliant on client updates, leading to performance volatility when client activity suddenly changes. For instance, FedAvg updates the global model by averaging client model parameters. This can result in instability when client activity changes, particularly in highly heterogeneous data distributions. FedProx introduces a regularization term to balance the weights between client and global models. This reduces fluctuation compared to FedAvg, but still shows significant volatility, indicating that adding a regularization term alone is insufficient to fully resolve the issue. MOON integrates global and client model features, handling non-IID data but still experience performance oscillations with changing client activity. FedAvgDBE improves knowledge transfer to reduce domain bias but remains affected by client loss or rejoining, impacting model stability. In contrast, FedProto, GPFL, and FIDSUS demonstrate better robustness. FedProto reduces client discrepancies by passing abstract class prototypes. This effectively utilizes information among clients and lowers dependency on updates. GPFL enhances model stability by integrating global and local features, making the fusion process less sensitive to the loss of a few clients. FIDSUS evaluates client similarity through an affinity matrix and performs knowledge sharing and aggregation based on this, adapting well to client dynamics. It assigns personalized local feature extractors to each client, enabling effective feature extraction even when some clients are absent. By combining feature representations from different rounds during aggregation, FIDSUS addresses the forgetting problem in highly heterogeneous settings. This approach ensures stable accuracy during client loss or rejoining, avoiding significant fluctuations. FIDSUS trains a global classifier using these enriched feature representations, enhancing generalization and stability. As a result, it achieves the highest average testing accuracy of 97%. This performance is attributed to its improved federated optimization strategy and dynamic adjustment mechanisms, which effectively handling client dynamics and data heterogeneity.

1) Performance Comparison in High Complexity Small-Scale Scenario: Fig. 9 illustrates the impact of different CARs on the accuracy of FL strategies in a high-complexity scenario with 10 clients. In this challenging environment, only FedProto, GPFL, and FIDSUS maintain robustness. The other strategies experience significant performance drops when CAR is 0.5. Furthermore, even when CAR is 1, these strategies

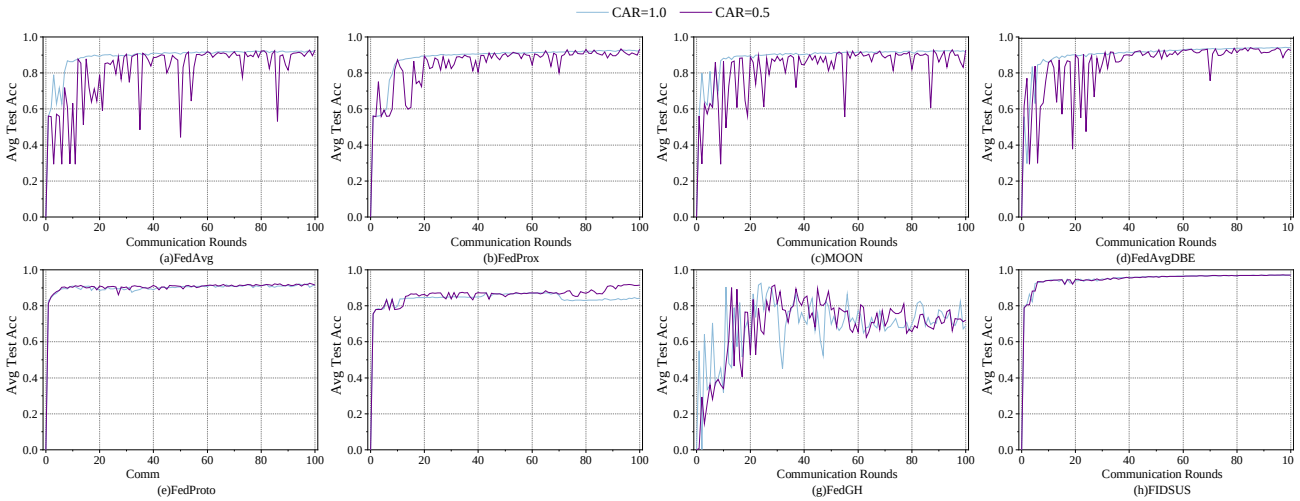


Fig. 8: Performance Comparison on the NSL-KDD Dataset in Dynamic Scenarios with 10 Clients

show a notable decrease in average accuracy compared to the low-complexity scenario. They also exhibit more frequent and larger fluctuations, underscoring their limitations in managing such environments. Among the three robust methods, FedProto fails to achieve an average testing accuracy above 80%. GPFL, after initially achieving high accuracy, starts to exhibit minor oscillations. However, these fluctuations have a minimal impact on overall performance. FIDSUS maintains considerable robustness and high accuracy, even in high-complexity scenarios. When CAR is 0.5, FIDSUS achieves a stable average testing accuracy of 88%, with only a 1% decrease compared to when CAR is 1. The FIDSUS global classifier enhances generalization, ensuring effective intrusion detection even in more complex scenarios. Its dynamic adjustment mechanism improves adaptability, allowing it to handle varying client conditions. By assessing client similarity through an affinity matrix, FIDSUS facilitates efficient knowledge sharing and aggregation. Each client uses a personalized feature extractor, ensuring accurate local data processing even when inactive. This design stabilizes accuracy during client fluctuations, allowing FIDSUS to maintain high performance in complex environments.

2) Performance Comparison in Low Complexity Medium-Scale Scenario: Fig. 10 illustrates the impact of different CARs on the performance of various FL strategies in a low-complexity scenario with 50 clients. As seen in previous analyses, FedProto, GPFL, and FIDSUS maintain high robustness, while FedGH continues to exhibit significant fluctuations under varying CAR settings. Although FedAvg, FedProx, MOON, and FedAvgDBE show some improvement compared to their performance in smaller-scale scenarios, they still experience fluctuations. This suggests that increasing the number of clients can help mitigate fluctuations caused by dynamic client changes. When comparing accuracy curves under different CAR parameters, it becomes clear that as the number of active clients decreases, fluctuations become more frequent and pronounced. Additionally, the number of active clients participating in FL can vary significantly depending on the scenario's scale. For instance, with a CAR of 0.5, a small-scale scenario may have only 5 active clients, while a moderate-scale scenario could have 25. A limited number of active clients can hinder the effective aggregation of model updates, whereas a

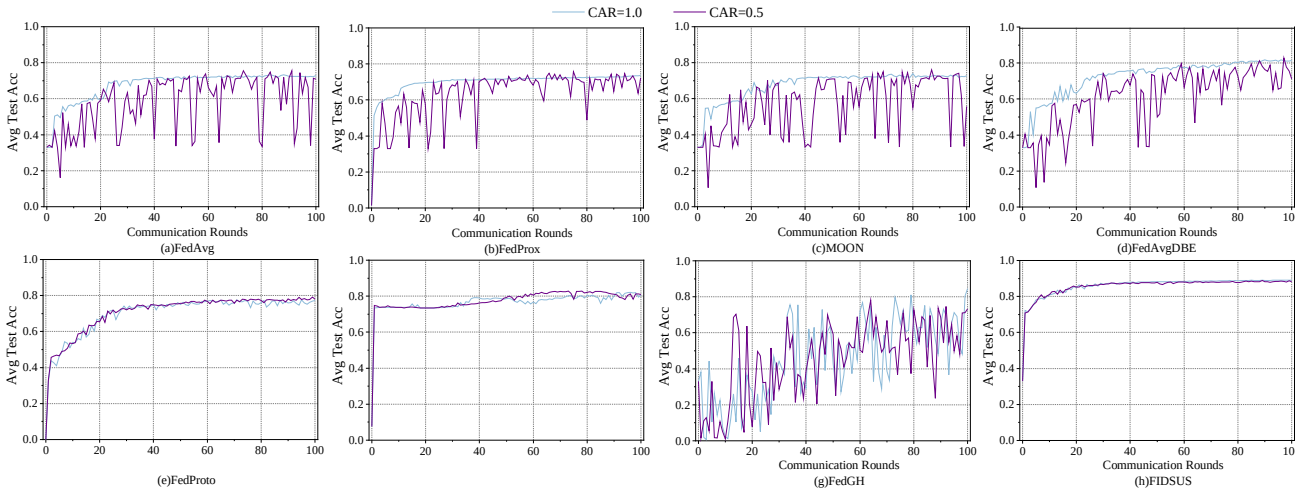


Fig. 9: Performance Comparison on the UNSW-NB15 Dataset in Dynamic Scenarios with 10 Clients

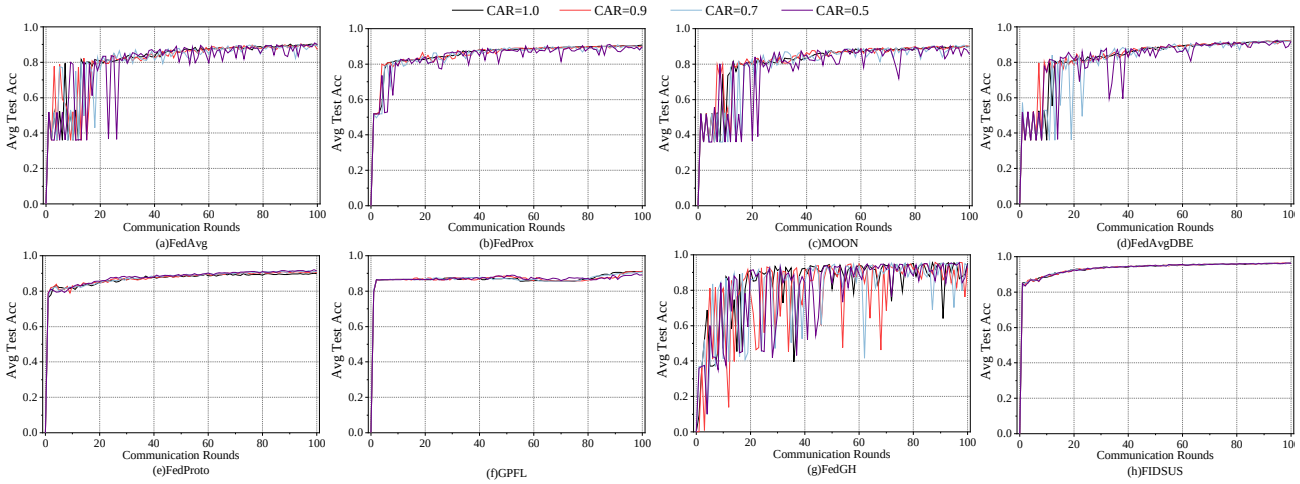


Fig. 10: Performance Comparison on the NSL-KDD Dataset in Dynamic Scenarios with 50 Clients

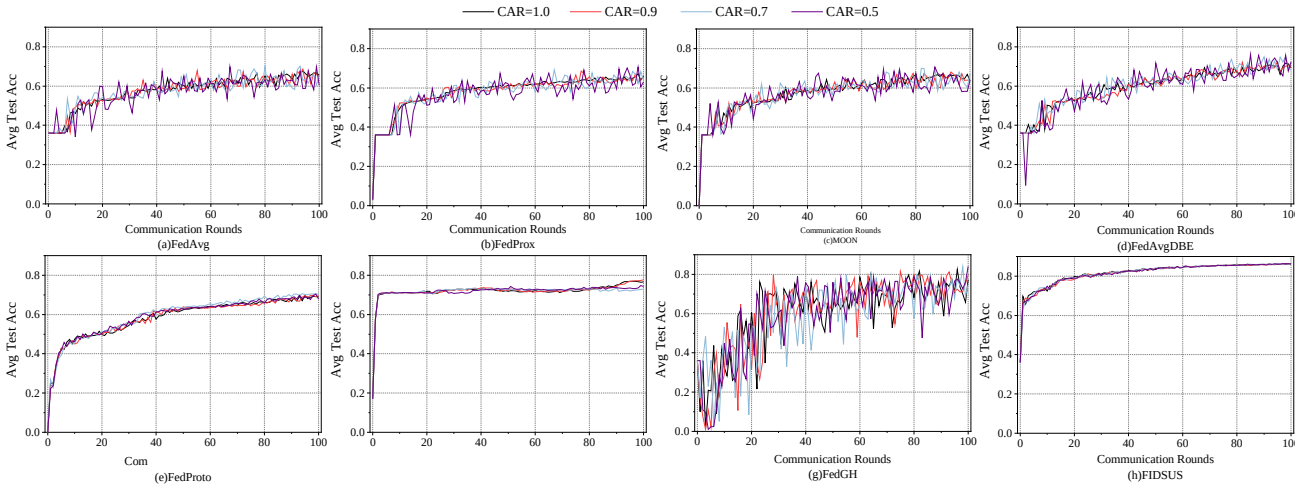


Fig. 11: Performance Comparison on the UNSW-NB15 Dataset in Dynamic Scenarios with 50 Clients

larger number of clients generally provides better conditions for robust FL. This pattern is also reflected in the performance of other strategies. Among the three robust methods, the accuracy curves under different CAR parameters overlap more in the moderate-scale scenario than in the small-scale one. This suggests that increasing the number of clients can help mitigate fluctuations caused by dynamic client changes. Notably, in this scenario, FIDSUS achieves the highest average testing accuracy of 96%. Its accuracy curves show the greatest overlap, indicating superior precision and robustness.

3) Performance Comparison in High Complexity Medium-Scale Scenario: Fig. 11 shows the impact of different CARs on the performance of various FL strategies in a high-complexity scenario with 50 clients. All federated strategies experience varying degrees of performance degradation, and their convergence speeds are also slowed. This highlights the challenges posed by high complexity and client variability in federated learning systems. Compared to the low-complexity scenario, the accuracy curves of FedAvg, FedProx, MOON, and FedAvgDBE exhibit greater fluctuations under varying CAR settings. Even when CAR is set to 1, there are noticeable fluctuations in the accuracy curves. None of these four methods achieve test accuracies above 70% in this scenario. While FedGH achieves relatively high accuracy, it still fails to address the significant fluctuations in performance under highly heterogeneous data. This issue stems from the global prediction head's heavy reliance on the quality of extracted features. Since its feature extractor does not effectively capture high-quality feature representations, it results in poor performance. FedProto, despite its higher robustness, achieves an average test accuracy of less than 70%. The GPFL strategy reaches a commendable test accuracy in early rounds but encounters a performance bottleneck later. In the final 10 rounds, the activity level of clients impacts its performance, with higher client activity yielding slightly better accuracy compared to lower activity. The proposed FIDSUS still achieves the highest average test accuracy of 86%. Furthermore, the accuracy curves for different CAR settings nearly overlap in the latter half of the rounds, demonstrating its superior robustness. With its unique optimization strategy and dynamic adjustment mechanism, FIDSUS excels in high-complexity scenarios with a moderate client scale. It showcases the highest average test accuracy and maintains stable robustness.

Based on the results and analysis, the proposed FIDSUS demonstrates superior accuracy and robustness across various complexities and scales of dynamic scenarios. Specifically, FIDSUS effectively mitigates the challenges posed by client dynamics and highly heterogeneous data through multi-level optimization strategies, including similarity-based knowledge sharing, and historical feature integration. To address client heterogeneity, FIDSUS evaluates client similarity using an affinity matrix constructed from feature distribution patterns. This enables dynamic selection of highly relevant clients for collaborative training, ensuring that knowledge sharing and model aggregation remain effective even under non-IID conditions. As a result, FIDSUS maintains stable performance and prevents the degradation of model accuracy when clients are lost or rejoin the system. Furthermore, FIDSUS adopts a dual-phase feature aggregation strategy, integrating feature representations both before and after updates. This method preserves historical feature distributions while incorporating newly learned patterns, effectively mitigating catastrophic forgetting. This is particularly beneficial in high-complexity environments, where abrupt shifts in data distribution can negatively impact conventional FL models. By leveraging these high-quality feature representations, FIDSUS enhances the training of the global classifier, leading to a more robust and generalizable model. Our experiments show that FIDSUS improves accuracy by 3% to 34% across various dynamic scenarios. Notably, even under varying client participation rates, FIDSUS maintains stable accuracy, demonstrating its ability to adapt to real-world deployment conditions.

F. Time Overheads Analysis

In FL, training time overhead impacts the practical application and deployment of models. Table IV presents the training time overhead for various FL strategies over 100 communication rounds. FedAvg and FedProx exhibit relatively

TABLE IV: Comparison of Time Overheads Across Different Strategies (Communication Rounds = 100)

Method	NSL-KDD			UNSW-NB15		
	NC=10	NC=50	NC=100	NC=10	NC=50	NC=100
FedAvg	108.43s	461.61s	873.73s	186.16s	812.41s	1880.48s
FedProx	115.43s	497.24s	926.77s	215.97s	774.79s	1991.86s
MOON	187.14s	995.05s	1861.47s	384.77s	1569.32s	3327.17s
FedAvgDBE	125.73s	582.50s	1093.16s	242.82s	984.09s	2069.76s
FedProto	213.44s	1043.92s	2549.03s	473.23s	2025.88s	5674.57s
GPFL	301.17s	1754.88s	3215.56s	584.53s	2419.57s	5602.82s
FedGH	164.38s	739.73s	1420.23s	315.83s	1315.73s	2899.38s
FIDSUS	**167.99s**	**843.76s**	**1684.40s**	**327.61s**	**1247.55s**	**3516.43s**

short runtime, as their update mechanisms primarily involve weighted averaging of client model parameters or the addition of regularization terms. These methods have lower computational complexity, resulting in shorter runtimes. However, their update mechanisms may show lower robustness and performance when facing highly heterogeneous data and dynamic client conditions. MOON and FedAvgDBE have longer runtimes compared to FedAvg and FedProx due to the inclusion of additional feature representation and knowledge transfer mechanisms. For instance, MOON handles non-IID data by merging global and client model features, while FedAvgDBE enhances knowledge transfer to reduce domain discrepancies. These additional steps increase computational complexity, leading to longer runtimes. FedProto incurs a longer runtime. This is primarily due to the extensive feature extraction and matching operations involved. These operations are necessary to reduce differences between client models using class prototypes. Despite its effectiveness in minimizing information disparity among clients, the increased computational load results in higher time consumption. GPFL also shows extended runtimes as it integrates global and local features to enhance model stability. This process involves multiple feature extraction and fusion operations, increasing the computational burden. Although GPFL has advantages in maintaining performance amid client activity variations, it requires additional computation time to sustain stability. FedGH falls between these strategies in terms of time consumption. Its global prediction head relies heavily on feature representations. This dependency makes the feature extraction process complex, resulting in a moderate time overhead. While it performs well in some scenarios, the complexity of its feature extractor limits its time efficiency. The proposed FIDSUS performs well in terms of runtime. Although its time consumption is slightly higher than FedAvg and FedProx, it maintains a good balance among all strategies. FIDSUS uses an affinity matrix to evaluate client similarity. This approach facilitates knowledge sharing and aggregation, effectively reducing unnecessary computations. Additionally, by combining historical and current features during aggregation, FIDSUS optimizes computational efficiency despite adding some computation steps. Overall, FIDSUS effectively controls computation time through its dynamic adjustment mechanism while ensuring high accuracy and robustness.

V. Conclusion

This study proposes FIDSUS, a federated intrusion detection framework designed for dynamic UAV swarm networks. By using an affinity matrix for dynamic client similarity assessment and efficient knowledge sharing, FIDSUS maintains high accuracy despite changing network conditions. It also mitigates forgetting by integrating historical and current feature representations, enhancing the model's generalization. Experimental results on the NSL-KDD and UNSW-NB15 datasets show FIDSUS achieves 4% to 34% higher accuracy than existing FL methods, demonstrating superior performance in accuracy, robustness, and stability, while maintaining reasonable time costs. In dynamic scenarios, FIDSUS ensures high detection accuracy and precision, making it a practical and effective solution for UAV network intrusion detection.

Despite the promising results, FIDSUS has some limitations. First, it assumes that UAV terminals can perform local computations efficiently, which may not hold in resource-constrained environments, limiting scalability in large UAV networks. Second, the use of an affinity matrix for client similarity adds extra communication overhead compared to other federated learning methods, potentially causing delays in large or dynamic networks. Additionally, while integrating historical and current features helps address the forgetting issue, it also increases computational costs. These finer-grained operations improve accuracy and robustness but come at the expense of higher computational complexity.

Future work will focus on addressing these limitations. We aim to improve scalability by exploring lightweight models and edge computing techniques. We will also optimize the affinity matrix to reduce communication overhead and investigate ways to balance performance and resource consumption. Finally, we plan to enhance FIDSUS's adaptability to extreme network conditions and long-term drift by incorporating reinforcement or incremental learning for better stability and responsiveness.

References

[1] H. Feng, Q. Li, W. Wang, A. K. Bashir, A. K. Singh, J. Xu, and K. Fang, "Security of target recognition for uav forestry remote sensing based on multi-source data fusion transformer framework," *Information Fusion*, vol. 112, p. 102555, 2024.

[2] G. Sun, L. He, Z. Sun, Q. Wu, S. Liang, J. Li, D. Niyato, and V. C. Leung, "Joint task offloading and resource allocation in aerial-terrestrial uav networks with edge and fog computing for post-disaster rescue," *IEEE Transactions on Mobile Computing*, 2024.

[3] T. Wang, J. Tian, K. Fang, T. R. Gadekallu, and W. Wang, "Ai and digital twin for consumer electronics in smart cities," *IEEE Consumer Electronics Magazine*, 2024.

[4] A. Sharma, P. Vanjani, N. Paliwal, C. M. W. Basnayaka, D. N. K. Jayakody, H.-C. Wang, and P. Muthuchidambaranathan, "Communication and networking technologies for uavs: A survey," *Journal of Network and Computer Applications*, vol. 168, p. 102739, 2020.

[5] J. Cai, T. Liu, T. Wang, H. Feng, K. Fang, A. K. Bashir, and W. Wang, "Multi-source fusion enhanced power-efficient sustainable computing for air quality monitoring," *IEEE Internet of Things Journal*, 2024.

[6] X. Zhou, X. Wen, Z. Wang, Y. Gao, H. Li, Q. Wang, T. Yang, H. Lu, Y. Cao, C. Xu *et al.*, "Swarm of micro flying robots in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm5954, 2022.

[7] J. John, K. Harikumar, J. Senthilnath, and S. Sundaram, "An efficient approach with dynamic multiswarm of uavs for forest firefighting," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2024.

[8] R. Greco, E. Barca, P. Raumonen, M. Persia, and P. Tartarino, "Methodology for measuring dendrometric parameters in a mediterranean forest with uavs flying inside forest," *International Journal of Applied Earth Observation and Geoinformation*, vol. 122, p. 103426, 2023.

[9] A. K. Sangaiah, F.-N. Yu, Y.-B. Lin, W.-C. Shen, and A. Sharma, "Uav t-yolo-rice: An enhanced tiny yolo networks for rice leaves diseases detection in paddy agronomy," *IEEE Transactions on Network Science and Engineering*, 2024.

[10] A. K. Sangaiah, J. Anandakrishnan, A. R. Devarapelly, M. L. A. B. Mohamad, G.-B. Bian, M. J. Alenazi, and S. A. AlQahtani, "R-uav-net: Enhanced yolov4 with graph-semantic compression for transformative uav sensing in paddy agronomy," *IEEE Transactions on Cognitive Communications and Networking*, 2024.

[11] Y. Tian, K. Liu, K. Ok, L. Tran, D. Allen, N. Roy, and J. P. How, "Search and rescue under the forest canopy using multiple uavs," *The International Journal of Robotics Research*, vol. 39, no. 10-11, pp. 1201–1221, 2020.

[12] X. Sun, D. W. K. Ng, Z. Ding, Y. Xu, and Z. Zhong, "Physical layer security in uav systems: Challenges and opportunities," *IEEE Wireless Communications*, vol. 26, no. 5, pp. 40–47, 2019.

[13] H. J. Hadi, Y. Cao, K. U. Nisa, A. M. Jamil, and Q. Ni, "A comprehensive survey on security, privacy issues and emerging defence technologies for uavs," *Journal of Network and Computer Applications*, vol. 213, p. 103607, 2023.

[14] D. He, S. Chan, and M. Guizani, "Communication security of unmanned aerial vehicles," *IEEE Wireless Communications*, vol. 24, no. 4, pp. 134–139, 2016.

[15] A. Altaweel, S. Aslam, and I. Kamel, "Security attacks in opportunistic mobile networks: A systematic literature review," *Journal of Network and Computer Applications*, p. 103782, 2023.

[16] H. Liao, M. Z. Murah, M. K. Hasan, A. H. M. Aman, J. Fang, X. Hu, and A. U. R. Khan, "A survey of deep learning technologies for intrusion detection in internet of things," *IEEE Access*, 2024.

[17] S. Dramé-Maigné, M. Laurent, L. Castillo, and H. Ganem, "Centralized, distributed, and everything in between: Reviewing access control solutions for the iot," *ACM Computing Surveys (CSUR)*, vol. 54, no. 7, pp. 1–34, 2021.

[18] K. Fang, T. Wang, L. Tong, X. Fang, Y. Pan, W. Wang, and J. Li, "Non-intrusive security assessment methods for future autonomous transportation iov," *IEEE Transactions on Automation Science and Engineering*, 2023.

[19] Z. Qin, G. Y. Li, and H. Ye, "Federated learning and wireless communications," *IEEE Wireless Communications*, vol. 28, no. 5, pp. 134–140, 2021.

[20] Y. Ding, Z. Yang, Q.-V. Pham, Y. Hu, Z. Zhang, and M. Shikh-Bahaei, "Distributed machine learning for uav swarms: Computing, sensing, and semantics," *IEEE Internet of Things Journal*, 2023.

[21] R. Ye, Z. Ni, C. Xu, J. Wang, S. Chen, and Y. C. Eldar, "Fedfm: Anchor-based feature matching for data heterogeneity in federated learning," *IEEE Transactions on Signal Processing*, 2023.

[22] Y. Qu, H. Dai, Y. Zhuang, J. Chen, C. Dong, F. Wu, and S. Guo, "Decentralized federated learning for uav networks: Architecture, challenges, and opportunities," *IEEE Network*, vol. 35, no. 6, pp. 156–162, 2021.

[23] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[24] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and trends® in machine learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[25] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 9, pp. 3400–3413, 2019.

[26] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 723–773, 2012.

[27] T. Li, J. Zhang, M. S. Obaidat, C. Lin, Y. Lin, Y. Shen, and J. Ma, "Energy-efficient and secure communication toward uav networks," *IEEE Internet of Things Journal*, vol. 9, no. 12, pp. 10 061–10 076, 2021.

[28] H. Xie, J. Zheng, T. He, S. Wei, C. Shan, and C. Hu, "B-uavm: A blockchain-supported secure multi-uav task management scheme," *IEEE Internet of Things Journal*, vol. 10, no. 24, pp. 21 240–21 253, 2023.

[29] R. Ye, Y. Peng, F. Al-Hazemi, and R. Boutaba, "A robust cooperative jamming scheme for secure uav communication via intelligent reflecting surface," *IEEE Transactions on Communications*, 2023.

[30] G. Folino and P. Sabatino, "Ensemble based collaborative and distributed intrusion detection systems: A survey," *Journal of Network and Computer Applications*, vol. 66, pp. 1–16, 2016.

[31] G. D. L. T. Parra, P. Rad, K.-K. R. Choo, and N. Beebe, "Detecting internet of things attacks using distributed deep learning," *Journal of Network and Computer Applications*, vol. 163, p. 102662, 2020.

[32] X. Zhao, G. Huang, L. Gao, M. Li, and Q. Gao, "Low load dids task scheduling based on q-learning in edge computing environment," *Journal of Network and Computer Applications*, vol. 188, p. 103095, 2021.

[33] M. S. Mousa'B, M. K. Hasan, R. Sulaiman, S. Islam, and A. U. R. Khan, "An explainable ensemble deep learning approach for intrusion detection in industrial internet of things," *IEEE Access*, vol. 11, pp. 115 047–115 061, 2023.

[34] F. Tlili, S. Ayed, and L. C. Fourati, "Exhaustive distributed intrusion detection system for uavs attacks detection and security enforcement (e-dids)," *Computers & Security*, vol. 142, p. 103878, 2024.

[35] L. You, J. He, W. Wang, and M. Cai, "Autonomous transportation systems and services enabled by the next-generation network," *IEEE Network*, vol. 36, no. 3, pp. 66–72, 2022.

[36] H. Chen, H. Wang, Q. Long, D. Jin, and Y. Li, "Advancements in federated learning: Models, methods, and privacy," *ACM Computing Surveys*, 2023.

[37] X. He, Q. Chen, L. Tang, W. Wang, and T. Liu, "Cgan-based collaborative intrusion detection for uav networks: A blockchain-empowered distributed federated learning approach," *IEEE Internet of Things Journal*, vol. 10, no. 1, pp. 120–132, 2022.

[38] H. J. Hadi, Y. Cao, S. Li, Y. Hu, J. Wang, and S. Wang, "Real-time collaborative intrusion detection system in uav networks using deep learning," *IEEE Internet of Things Journal*, 2024.

[39] W. Wang, O. Abbasi, H. Yanikomeroglu, C. Liang, L. Tang, and Q. Chen, "A vertical heterogeneous network (vhetnet)–enabled asynchronous federated learning-based anomaly detection framework for ubiquitous iot," *IEEE Open Journal of the Communications Society*, 2023.

[40] Z. Zhang, Y. Zhang, H. Li, S. Liu, W. Chen, Z. Zhang, and L. Tang, "Federated continual representation learning for evolutionary distributed intrusion detection in industrial internet of things," *Engineering Applications of Artificial Intelligence*, vol. 135, p. 108826, 2024.

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

[42] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE symposium on computational intelligence for security and defense applications*. Ieee, 2009, pp. 1–6.

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

[44] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine learning and systems*, vol. 2, pp. 429–450, 2020.

[45] Q. Li, B. He, and D. Song, "Model-contrastive federated learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 10 713–10 722.

[46] J. Zhang, Y. Hua, J. Cao, H. Wang, T. Song, Z. Xue, R. Ma, and H. Guan, "Eliminating domain bias for federated learning in representation space," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[47] Y. Tan, G. Long, L. Liu, T. Zhou, Q. Lu, J. Jiang, and C. Zhang, "Fedproto: Federated prototype learning across heterogeneous clients," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 8, 2022, pp. 8432–8440.

[48] J. Zhang, Y. Hua, H. Wang, T. Song, Z. Xue, R. Ma, J. Cao, and H. Guan, "Gpfl: Simultaneously learning global and personalized feature information for personalized federated learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 5041–5051.

[49] L. Yi, G. Wang, X. Liu, Z. Shi, and H. Yu, "Fedgh: Heterogeneous federated learning with generalized global header," in *Proceedings of the 31st ACM International Conference on Multimedia*, 2023, pp. 8686–8696.

[50] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[51] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.