Consensus-Constrained Dual-Federated Optimization with Client-Side Dynamic Weighting for Distributed Mobile Threat Defense

Mohammad Mashreghi*, Mohammad Hossein Badiei*, and Md. Jalil Piran (Senior Member, IEEE)

***Abstract*—Mobile systems are increasingly facing advanced and complex malware threats. Detecting these threats while keeping user data private and ensuring scalability remains a key challenge. Federated learning (FL) helps by training models directly on devices without collecting user data. However, common FL methods like FedAvg assume that all clients are equally reliable, which isn't true in real-world conditions. These methods also fail to adapt to changing client performance and struggle when data is not evenly distributed or when only some clients participate in each round. In this article, we propose DW-FedADMM, a federated optimization framework that fuses primal-dual coordination with a dynamic client weighting mechanism to enhance learning efficiency and model robustness. Our method introduces dual variables to regularize local updates, mitigating divergence between client and global objectives, while dynamically adjusting aggregation weights based on per-round client performance. This dual-adaptive mechanism ensures both convergence stability and client-aware aggregation in asynchronous and communication-constrained environments. We conduct extensive evaluations on three benchmark Android malware datasets (e.g. Drebin, Malgenome, and Tuandromd) under diverse participation and communication scenarios. Experimental results indicate that DW-FedADMM achieves overall superior performance compared to representative baselines, including FedAvg, FedProx, DW-FedAvg, and DW-FedProx, across diverse classification tasks. FedADMM achieves up to 1.67% higher accuracy, 1.95% macro F1-score, 0.58% precision, 0.10% AUC, and 2.19% specificity compared to the mean performance of FedAvg, FedProx, DW-FedAvg, and DW-FedProx, with all metrics exceeding 90%, highlighting the strength of our approach. DW-FedADMM further enhances FedADMM by 0.23% in accuracy, 0.07% in precision, 0.60% in macro F1-score.**

***Index Terms*—Communication-efficient federated learning, Distributed cybersecurity, Primal-dual optimization, Dynamic client weighting, Mobile threat classification.**

I. Introduction

THE proliferation of Android devices has revolutionized mobile computing, offering users unprecedented access to applications and services. However, this ubiquity has also made Android a prime target for malicious actors. The open-source nature of the Android platform, coupled with its fragmented ecosystem, has led to a surge in malware attacks, ranging from spyware and ransomware to sophisticated Advanced Persistent Threats (APTs) [1]. Traditional malware detection approaches, which often rely on centralized data collection and analysis, face significant challenges in this context. These methods not only raise privacy concerns but also struggle with scalability and adaptability to the diverse and evolving threat landscape [2].

To address these challenges, federated learning (FL) has emerged as a promising paradigm. As depicted in Fig. 1, FL enables decentralized model training across multiple devices, allowing each client to train on its local data and share only model updates with a central server [3]. As illustrated, the training proceeds iteratively: in step ①, the server either initializes the global model (at the first round) or aggregates updates from previous clients. In step ②, this model is broadcast to a randomly selected subset of clients. In step ③, each participating client updates the received model locally using its private data and partial optimization. Finally, in step ④, the updated client models are sent back to the server and integrated to refine the global model. This process is repeated until convergence or stabilization.

However FL, while promising for privacy-preserving and decentralized training across distributed mobile environments, introduces several inherent challenges that differentiate it from conventional distributed learning paradigms. Chief among these is the communication overhead incurred during frequent model synchronization, which becomes particularly pronounced in bandwidth-constrained or large-scale settings where devices such as smartphones possess limited network and processing capabilities [4]. Furthermore, FL exhibits susceptibility to convergence instability, especially under data heterogeneity and client variance, which can degrade global model performance and delay optimization [5].

Another key issue is the heterogeneous nature of edge devices, where disparate computation capacities and intermittent connectivity create asynchronous update behaviors. This phenomenon, often termed the "straggler effect," introduces latency and temporal inconsistency during model aggregation [6]. Adding to this is stochastic client availability, in which only a subset of nodes can participate in any given training round, which requires algorithms that are inherently resilient to partial participation and asynchronous scheduling [7].

In response, several strategies have been proposed to mitigate these issues to enhance the robustness and reliability of malware classification in decentralized environments [8], [9]. For instance, one stream of research has explored adversarial-resilient strategies, such as integrating generative adversarial networks (GANs) or worst-case robust optimization formulations to mitigate data poisoning and model manipulation attacks [10], [11]. However, these methods face significant limitations in federated settings. GAN-based defenses often suffer from mode collapse, limited perturbation diversity, and unstable training dynamics, making them ill-suited for FL scenarios with resource-constrained clients [12], [13].

Mohammad Mashreghi and Mohammad H. Badie are with the Department of Electrical and Computer Engineering, University of Tehran, Tehran 1417614411, Iran, (email:m.mashreghi@ut.ac.ir; mho.badiei@gmail.com).

M. J. Piran is with the Department of Computer Science and Engineering, Sejong University, Seoul 05006, South Korea, e-mail: piran@sejong.ac.kr

*** These authors contributed equally to this work.**

Fig. 1: Overview of Federated Learning Framework

Similarly, robust optimization techniques typically rely on pessimistic upper bounds or adversarial sampling assumptions that may not hold in non-stationary malware distributions, ultimately compromising accuracy and imposing substantial computational burden [14].

Complementary to adversarial defenses, aggregation schemes resilient to Byzantine faults, such as Krum, coordinate-wise median, or geometric median approaches, have also been studied [15], [16]. While theoretically robust, these methods present scalability limitations. For instance, Krum exhibits quadratic computational complexity with respect to the number of clients, rendering it inefficient in large-scale deployments [17]. Median-based filters, although computationally lighter, exhibit reduced effectiveness in high-dimensional model spaces where distinguishing malicious updates from benign ones becomes increasingly difficult. Moreover, the geometric median entails iterative solvers, adding considerable latency and communication cost [18].

Additionally, FL-based malware classifiers have frequently relied on deep neural networks (DNNs), especially convolutional neural networks (CNNs), due to their capacity to extract hierarchical representations from complex input modalities [9], [19], [20]. These models have shown efficacy in leveraging semantic patterns from permission requests, API sequences, and network behaviors to distinguish between benign and malicious Android applications. Nevertheless, CNN-based architectures are inherently computationally expensive, making them impractical for inference or training on devices with constrained CPU and memory footprints [21]. Their performance is also tightly coupled with the availability of large, diverse datasets; a condition rarely met in federated environments characterized by non-IID, skewed, and imbalanced local data distributions [22], [23]. Furthermore, their opaqueness and lack of explainability hinder interpretability and trust in security-sensitive applications. The need for continuous adaptation to novel malware families further compounds bandwidth and computational demands.

On the other hand, primal-dual optimization methods, such as the Alternating Direction Method of Multipliers (ADMM), have been adapted for FL to address challenges like data heterogeneity by coordinating updates between clients and the server [24]. While effective in improving convergence and stability, these methods introduce complexities, including the need for synchronizing dual variables, which requires additional memory and update steps [25]. Moreover, the augmented Lagrangian structure necessitates careful tuning of penalty parameters to prevent oscillatory convergence or divergence, particularly in non-convex settings [26].

In parallel, Performance-based weighting strategies, such as Dynamic Weighted Federated Averaging (DW-FedAvg), prioritize high-quality client updates to enhance model robustness against data skew and participation variability [27]. However, these methods can exhibit instability when local evaluation metrics are noisy, biased, or unrepresentative of global trends. Overweighting certain clients may result in overfitting to local distributions or amplify variance in the global update, particularly under non-stationary data and fluctuating participation.

Despite substantial progress, FL in mobile environments remains challenged by systemic limitations, including high communication costs, unstable convergence under partial participation, and limited device capabilities. These issues are compounded by asynchronous updates, straggler effects, and fluctuating client availability, which disrupt coordination and slow global optimization. While prior solutions address specific aspects in isolation, a unified and scalable framework that ensures efficiency, adaptability, and optimization stability remains largely absent

To overcome the limitations of fragmented optimization and unstable aggregation in federated malware detection, we propose Dual-Federated Optimization with Dynamic Weighting (DW-FedADMM); a framework that orchestrates dual-informed coordination with performance-adaptive client prioritization. By embedding dual variables into a consensus-constrained optimization objective, the method enforces regularized alignment between local and global updates, mitigating inter-client inconsistency. Concurrently, client contributions are modulated through a soft-weighted aggregation scheme driven by validation-informed metrics, which interact with the dual dynamics to stabilize updates and suppress variance. This synergistic coupling of primal-dual regulation and adaptive weighting yields consistent improvements in convergence speed and generalization. Empirical evaluation across diverse benchmark datasets (e.g. Drebin, Malgenome, and Tuandromd) demonstrates that FedADMM, and particularly DW-FedADMM, achieves superior performance, with FedADMM showing improvements of up to 1.67% in accuracy, 1.95% in macro F1-score, 0.58% in precision, 0.10% in AUC, and 2.19% in specificity relative to the mean performance of FedAvg, FedProx, DW-FedAvg, and DW-FedProx, underscoring the effectiveness of our method. DW-FedADMM further enhances these results, yielding additional gains of 0.23% in accuracy, 0.07% in precision, and 0.60% in macro F1-score compared to FedADMM.

The source code for our experiments is publicly available on GitHub.

The rest of the paper is organized as follows: Section II details the proposed framework. Section III presents our experimental setup, results, and analysis. Finally, Section IV concludes the paper and outlines directions for future research.

II. Methodology

A. Background

To motivate the formulation of our proposed framework, we revisit the foundational components underlying federated mobile threat defense in Android ecosystems, emphasizing both system-level characteristics and optimization-theoretic principles.

1) Android Application Structure: Android applications are distributed as APK packages that encapsulate critical execution logic and metadata. Each APK typically contains: (i) `AndroidManifest.xml`, specifying component declarations, runtime permissions, and app configuration; (ii) `classes.dex`, holding Dalvik Executable bytecode; and (iii) embedded resources (layouts, multimedia, and UI descriptors). These structured representations provide static and semantic features suitable for local inference. Within a federated setting, this modularity facilitates client-side feature extraction without sharing raw binaries, ensuring compliance with privacy constraints.

2) Federated Optimization Framework.: Consider n mobile clients indexed by $i \in \{1, \dots, n\}$, each holding a local dataset $\mathcal{D}_i$ with $|\mathcal{D}_i|$ samples. The objective is to collaboratively optimize a global model parameterized by $\bar{x} \in \mathbb{R}^d$ via:

$$\min_{\bar{x}\in\mathbb{R}^d} \sum_{i=1}^{n} \frac{|\mathcal{D}_i|}{|\mathcal{D}|} \mathbb{E}_{z\sim\mathcal{D}_i} \left[\mathcal{L}(h_{\bar{x}}(z))\right], \tag{1}$$

where $\mathcal{L}$ is a sample-wise loss function (e.g., cross-entropy), and $h_{\bar{x}}$ denotes the model with parameters $\bar{x}$. In classical FedAvg, each client solves:

$$x_i^k \leftarrow \mathrm{SGD}^E\left(\bar{x}^{k-1}; \mathcal{D}_i\right), \quad \text{followed by} \quad \bar{x}^k \leftarrow \frac{1}{n}\sum_{i=1}^{n} x_i^k, \tag{2}$$

where E is the number of local epochs per round k. Despite its simplicity, FedAvg suffers in heterogeneous environments due to unconstrained client deviation.

To address this, FedProx augments the local objective with a proximal regularization:

$$x_i^k = \arg\min_{x\in\mathbb{R}^d} \left\{\mathcal{L}_i(x) + \frac{\mu}{2}\|x - \bar{x}^{k-1}\|^2\right\}, \tag{3}$$

where $\mu > 0$ controls proximity to the global reference point $\bar{x}^{k-1}$. This penalization mitigates client drift, especially under heterogeneous settings, and enhances convergence stability.

3) Federated Mobile Threat Defense: Leveraging federated optimization for Android threat defense enables scalable threat modeling without centralizing sensitive behavioral data. Clients extract high-dimensional representations from static code features, dynamic system traces, permission graphs, and execution profiles. These vectors inform distributed training of a mobile threat classifier. However, federated deployment introduces unique complications: (i) communication bottlenecks due to iterative synchronization, (ii) statistical heterogeneity across local datasets, and (iii) temporal asynchrony from fluctuating client availability. These issues impair convergence and motivate the development of dual-informed and adaptive aggregation mechanisms, as explored in this work.

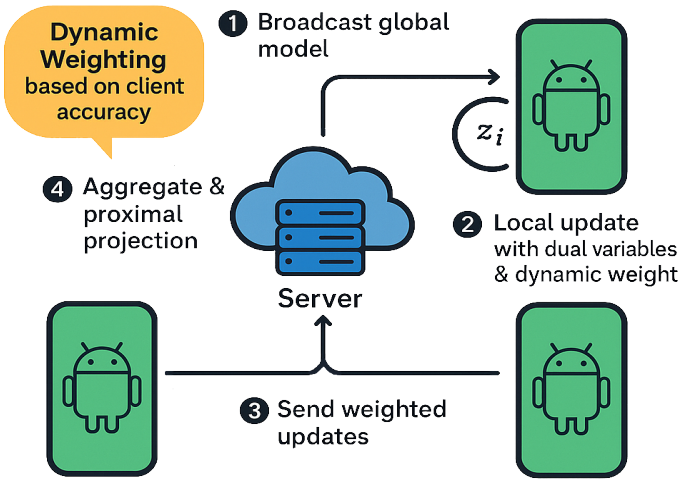

Fig. 2: Overview of DW-FedADMM framework for Federated Android Malware Defense

B. Proposed Method

In this subsection, we introduce DW-FedADMM, a dual-informed and adaptively weighted primal-dual framework for federated threat detection. As depicted in Fig. 2, our proposed algorithm encodes dual coupling, augmented Lagrangian regularization, and performance-driven interpolation into a unified optimization rule, enabling stable and variance-aware training under partial participation.

1) Problem Decomposition: We consider a federated mobile threat detection scenario, where a population of n Android devices collaboratively train a malware classifier without sharing raw data. Each device $i \in [n]$ possesses private data $\mathcal{D}_i$, extracted from application-level features such as permissions, API calls, and runtime behavior. The goal is to jointly optimize a global classifier $x \in \mathbb{R}^d$ that distinguishes between malicious and benign applications.

In this decentralized context, each client defines a local objective $f_i(x)$, typically representing a cross-entropy loss over its labeled malware samples. The total federated objective becomes:

$$\min_x \; F(x) = \sum_{i=1}^{n} \frac{|\mathcal{D}_i|}{|\mathcal{D}|} f_i(x). \tag{4}$$

To preserve personalization and local adaptability, DW-FedADMM introduces separate local decision variables x_i and enforces consensus via the constraint $x_i = \bar{x}$ for all i.

2) Augmented Lagrangian Formulation: DW-FedADMM adopts a primal-dual formulation through augmented Lagrangian methods. The constrained problem:

$$\min_{\{x_i\},\bar{x}} \sum_{i=1}^{n} f_i(x_i), \quad \text{s.t.} \quad x_i = \bar{x}, \; \forall i, \tag{5}$$

is relaxed by introducing dual variables z_i and an augmentation term to penalize violations of consensus. The per-client augmented Lagrangian is:

$$\mathcal{L}_i(x_i, \bar{x}, z_i) = f_i(x_i) + \langle z_i, x_i - \bar{x}\rangle + \frac{\gamma}{2}\|x_i - \bar{x}\|^2, \tag{6}$$

Algorithm 1 DW-FedADMM

Initialize $x^0, \gamma > 0, K$, and tolerances $\epsilon_{i,0} (i \in [n])$
Initialize the server with $\bar{x}^0 = x^0$
Initialize all clients with $z_i^0 = 0$, and $x_i^0 = \hat{x}^0 = x^0$
for $k = 0, 1, \ldots, K$ **do**
  Randomly sample $S_k \subseteq [n]$ with size S *// Select clients randomly each round*
  ▷ **Client side:**
  for each client $i \in S_k$ **do**
    Receive $\bar{x}^k$ and $BestAcc_k$ from the server
    Compute $\omega_{i,k+1} \leftarrow \frac{Acc_{i,prev}}{BestAcc_k}$ *// Dynamic weight coefficient*
    Update local model: $\hat{x_i}^{k+1} \leftarrow \omega_{i,k+1} \cdot \hat{x}^k + (1 - \omega_{i,k+1}) \cdot \hat{x_i}^k$
    $x_i^{k+1} \approx \arg\min_{x_i} \mathcal{L}_i\big(x_i, \bar{x}^k, z_i^k\big)$
    $z_i^{k+1} = z_i^k + \gamma\left(x_i^{k+1} - \bar{x}^k\right)$ *// Dual update*
    $\hat{x_i}^{k+1} = x_i^{k+1} - \frac{1}{\eta} z_i^{k+1}$
    Send $\hat{x_i}^{k+1}$ and $Acc_{i,k}$ to the server
  end for
  ▷ **Server side:**
  $BestAcc_k \leftarrow \max(\{Acc_{i,k}\}_{i \in S_k})$
  $\bar{x}^{k+1} = \mathrm{prox}_{\gamma/n}\big(\hat{x}^{k+1}\big)$
end for

where $\gamma > 0$ controls the penalty strength. This formulation enables each client to perform a local descent with a quadratic regularizer, followed by a dual ascent step.

3) Dynamic Weight Modulation: Another component of DW-FedADMM is the dynamic weighting strategy, which adaptively modulates each client's influence during aggregation based on recent predictive performance. For each round k, every participating client i computes a weight:

$$\omega_{i,k+1} = \frac{\mathrm{Acc}_{i,\mathrm{prev}}}{\mathrm{BestAcc}_k}, \tag{7}$$

Where $\mathrm{Acc}_{i,\mathrm{prev}}$ represents the accuracy result of client i from the most recent round in which it was selected, and $\mathrm{BestAcc}_k$ denotes the highest accuracy achieved among all sampled clients.

This coefficient is used to generate an interpolated state before local training:

$$\hat{x}_i^{k+1} = \omega_{i,k+1} \cdot \hat{x}^k + (1 - \omega_{i,k+1}) \cdot \hat{x}_i^k, \tag{8}$$

allowing smoother transitions and mitigating divergence from underperforming clients. This interpolation acts as an implicit regularizer, reducing gradient variance and improving global model stability.

4) Optimization Protocol: The following outlines the complete iterative process of DW-FedADMM in a step-by-step manner. Let the global learning process evolve over K communication rounds. At each round k, the system performs the following operations:

- **Initialization.** The server initializes the global model state as $\bar{x}^0 = x^0$, and each client $i \in [n]$ is initialized with $x_i^0 = x^0$, dual variable $z_i^0 = 0$, and interpolated proxy $\hat{x}_i^0 = x^0$.
- **Step 1: Client Sampling.** At each round k, a subset of clients $S_k \subseteq [n]$ is randomly selected to participate. The cardinality $|S_k| = S$ defines the participation rate.
- **Step 2: Broadcast.** The server broadcasts the current global model $\bar{x}^k$ and the best previous client accuracy $\mathrm{BestAcc}_k$ to all sampled clients.
- **Step 3: Local Computation (Per Client $i \in S_k$).** Each participating client executes the following operations:
  - **Dynamic Weight Retrieval.** The client fetches or recomputes its interpolation coefficient $\omega_{i,k+1}$ using (7), reflecting its most recent predictive performance relative to the best-performing peer.
  - **Interpolated Initialization.** A smoothed local initialization $\hat{x}_i^{k+1}$ is computed via (8), combining global consistency from $\hat{x}^k$ with local inertia from $\hat{x}_i^k$.
  - **Primal Update.** The client solves the augmented Lagrangian subproblem, as depicted in (6), to obtain an updated local model x_i^{k+1}. This step typically involves gradient-based minimization using local data $\mathcal{D}_i$.
  - **Dual Update.** The dual variable is adjusted using the classic primal-dual ascent rule:
  $$z_i^{k+1} = z_i^k + \gamma\left(x_i^{k+1} - \bar{x}^k\right). \tag{9}$$
  - **Communication-Efficient Projection.** The client compresses the update via dual-adjusted correction:
  $$\hat{x}_i^{k+1} = x_i^{k+1} - \frac{1}{\eta} z_i^{k+1}, \tag{10}$$
  effectively encoding both primal and dual signals.
  - **Transmission.** The tuple $\left(\hat{x}_i^{k+1}, \mathrm{Acc}_{i,k}\right)$ is sent to the server for global aggregation and update.
- **Step 4: Server Aggregation.**
  - **Performance-Guided Normalization.** Compute the new best accuracy:
  $$\mathrm{BestAcc}_k = \max\{\mathrm{Acc}_{i,k} : i \in S_k\}. \tag{11}$$
  - **Model Averaging.** Aggregate the updates from sampled clients:
  $$\hat{x}^{k+1} = \frac{1}{|S_k|} \sum_{i \in S_k} \hat{x}_i^{k+1}. \tag{12}$$
  - **Proximal Projection.** Update the global model using a proximal operator:
  $$\bar{x}^{k+1} = \mathrm{prox}_{\gamma/n}(\hat{x}^{k+1}), \tag{13}$$
  where the proximal mapping may encode global regularization (e.g., ℓ_2-norm penalty).

This iterative protocol allows DW-FedADMM to balance local expressiveness and global coordination while adapting client contributions based on performance feedback, ensuring stable and accelerated convergence under data distributions and intermittent client participation.

III. Simulation Results and Discussion

To assess the performance of our proposed method, we conduct extensive simulations across a variety of configurations. These experiments are designed to evaluate performance, convergence behavior, communication efficiency, and scalability in comparison to established FL baselines.

A. Dataset Description

To evaluate the proposed approach, we utilize three benchmark datasets for Android malware detection: Malgenome, Drebin, and Tunadromd. Each dataset is described below:

- **Malgenome.** This dataset, derived from the Android Malware Genome Project [28], comprises 3,799 Android application samples, including 2,539 benign and 1,260 malicious applications. It encompasses a total of 215 features extracted from the samples.
- **Drebin.** Sourced from the Drebin Project [29], this dataset contains 15,036 application samples, with 9,476 classified as benign and 5,560 as malicious. The dataset includes 215 features, providing a comprehensive representation of Android application characteristics.
- **Tunadromd.** As described in [30], this dataset consists of 4,465 application samples, including 903 benign and 3,565 malicious samples. It incorporates a total of 241 features, offering a diverse set of attributes for malware analysis.

B. Experimental Setup

This subsection outlines the experimental framework of our study, including machine configuration, software environment, classifier architecture, optimization strategy, and parameter settings. Details are provided below:

1) Machine Configuration: The experiments were conducted on Kaggle's cloud-based computational environment, utilizing a 64-bit operating system with up to 4 CPU cores and 30 GB of RAM, suitable for deep learning tasks and federated learning experiments.

2) Base Classifier and Optimizer: The base classifier is a four-layer feed-forward neural network designed for binary classification (benign vs. malicious applications). This architecture was selected for two primary reasons: (1) to facilitate rapid training and efficient communication among clients in a FL framework, thereby reducing latency during model updates, and (2) to ensure a compact model suitable for deployment on resource-constrained mobile devices. The network architecture comprises:

1) *First Hidden Layer*: 200 neurons, capturing a wide range of input features.
2) *Second Hidden Layer*: 100 neurons, refining feature representations.
3) *Third Hidden Layer*: 50 neurons, compressing the feature space for efficient classification.
4) *Output Layer*: A single neuron with a sigmoid activation function, producing probabilities for binary classification (benign or malicious).

The neural network architecture was optimized through empirical iterative testing, as no standardized automated method exists for this context. Hidden layers employ ReLU activation to ensure non-linearity and address vanishing gradient challenges, while the sigmoid activation in the output layer supports effective binary classification. The SGD optimizer was selected for its simplicity, efficiency, and suitability for distributed updates in FL.

3) Training Parameters: The models were trained using a partial FL approach, where 80% of available clients are randomly selected in each training round to participate and aggregate their updates. The training process was configured with the following hyperparameters:

- *Batch Size*: 32, balancing computational efficiency and gradient stability.
- *Local Epochs*: 32, fixed for each client during training.
- *Cross-Validation*: An 80-20 hold-out strategy, with 80% of the dataset used for training and 20% for testing.
- *Learning Rate*: 0.0001 for the SGD optimizer, ensuring steady convergence without overshooting.
- *Client Selection*: 0.8, specifying the ratio of clients selected for training in each round.

The experimental setup ensures that our DW-FedADMM method for Android malware defense is both practical to implement and rigorously evaluated. This robust framework provides a solid foundation for the performance results presented in subsequent sections.

C. Discussion and Analysis

This subsection analyzes the performance metrics of our proposed method via simulations evaluating performance, convergence, communication efficiency, and scalability against federated learning (FL) baselines. Results are presented in Table I (accuracy for 10 and 20 rounds) and Table II (specificity, AUC, and F1 scores for 10 and 20 rounds). Each table lists datasets (Drebin, Malgenome, Tuandromd), client counts (5, 10, 15), and six FL algorithms (FedAvg, FedProx, FedADMM, DW-FedAvg, DW-FedProx, DW-FedADMM), with top metrics in bold. Fig. 3 visualizes overall mean accuracy across datasets, summarizing algorithm performance.

Drebin Dataset. The Drebin dataset demonstrates stable performance across all metrics, with DW-FedADMM consistently performing well. In Table II (10 rounds), DW-FedADMM achieves the highest F1 score with 15 clients (96.77%) and is competitive with 5 clients (97.51%, led by DW-FedAvg at 97.58%). For 10 clients, FedADMM leads with an F1 score of 97.29%. AUC scores exceed 99.4%, and specificity remains high, though it slightly decreases with more clients (e.g., 97.29% for FedAvg with 5 clients to 95.53% with 15 clients). In terms of accuracy (Table I), DW-FedAvg leads for 5 clients (98.22%) and FedADMM for 10 clients (98.00%), while DW-FedADMM excels for 15 clients (97.62%).

For 20 rounds (Table II), DW-FedADMM achieves the highest F1 score for 15 clients (96.78%) and performs strongly for 5 clients (97.69%, led by DW-FedAvg) and 10 clients (97.28%, led by FedADMM at 97.34%). Accuracy metrics (Table I) show DW-FedAvg leading for 5 clients (98.30%),

TABLE I: Global Models Results for Accuracy Across Datasets and Algorithms for 10 and 20 Rounds (in %)

	No. of Clients	10 Rounds						20 Rounds					
		FedAvg	**FedProx**	**FedADMM**	**DW-FedAvg**	**DW-FedProx**	**DW-FedADMM**	**FedAvg**	**FedProx**	**FedADMM**	**DW-FedAvg**	**DW-FedProx**	**DW-FedADMM**
Drebin	5	98.21 ±0.20	98.14 ±0.16	98.07 ±0.31	**98.22 ±0.20**	98.17 ±0.16	98.16 ±0.29	98.28 ±0.17	98.20 ±0.14	97.99 ±0.34	**98.30 ±0.17**	98.26 ±0.14	98.05 ±0.34
	10	97.63 ±0.32	97.63 ±0.32	**98.00 ±0.51**	97.63 ±0.42	97.62 ±0.44	97.90 ±0.53	97.77 ±0.26	97.75 ±0.27	**98.04 ±0.44**	97.76 ±0.30	97.77 ±0.26	97.99 ±0.50
	15	97.39 ±0.80	97.37 ±0.82	97.53 ±0.94	97.29 ±0.85	97.27 ±0.87	**97.62 ±1.01**	97.48 ±0.66	97.47 ±0.66	97.60 ±1.10	97.53 ±0.63	97.50 ±0.61	**97.61 ±1.07**
Malgenome	5	98.28 ±2.46	98.22 ±2.57	98.93 ±0.44	99.00 ±0.53	98.92 ±0.75	**99.21 ±0.57**	**98.93 ±0.95**	98.89 ±0.97	98.56 ±1.49	98.72 ±1.71	98.57 ±2.45	98.65 ±1.61
	10	94.53 ±10.02	94.38 ±10.07	95.37 ±9.89	91.95 ±13.06	91.92 ±13.06	**95.71 ±9.93**	96.99 ±7.29	96.98 ±7.29	**97.57 ±7.27**	97.16 ±7.21	96.95 ±7.16	97.44 ±7.24
	15	88.86 ±13.28	88.74 ±13.28	92.30 ±13.19	87.95 ±14.44	87.67 ±14.36	**95.29 ±9.85**	93.19 ±11.57	93.13 ±11.57	95.63 ±9.91	93.31 ±11.59	93.14 ±11.53	**95.81 ±9.97**
Tuandromd	5	96.29 ±4.94	96.29 ±4.94	**98.09 ±1.24**	96.75 ±4.50	96.73 ±4.58	97.64 ±2.49	97.68 ±3.40	97.62 ±3.63	98.05 ±2.42	97.49 ±3.70	97.44 ±3.69	**98.59 ±0.46**
	10	92.81 ±6.78	92.50 ±7.03	**96.38 ±5.00**	93.61 ±6.22	93.36 ±6.22	96.34 ±5.05	94.18 ±6.36	94.13 ±6.34	96.55 ±5.04	95.00 ±5.74	94.98 ±5.75	**96.75 ±3.98**
	15	84.77 ±5.33	84.39 ±5.03	94.23 ±6.49	86.95 ±6.82	85.60 ±6.30	**94.34 ±6.53**	90.92 ±7.51	90.64 ±7.54	**96.29 ±4.98**	93.80 ±6.27	93.45 ±6.50	96.13 ±5.02

FedADMM for 10 clients (98.04%), and DW-FedADMM for 15 clients (97.61%). The slight improvement in accuracy from 10 to 20 rounds (e.g., 98.22% to 98.30% for DW-FedAvg with 5 clients) aligns with F1 score gains, indicating enhanced convergence, particularly for DW-FedADMM with larger client numbers.

Malgenome Dataset. The Malgenome dataset exhibits significant variability, particularly in specificity and F1 scores for larger client counts, with DW-FedADMM consistently outperforming others. In Table II (10 rounds), DW-FedADMM achieves the highest F1 scores across all client counts: 98.83% (5 clients), 88.71% (10 clients), and 88.06% (15 clients), with high standard deviations for 10 and 15 clients reflecting dataset heterogeneity. AUC scores remain above 99.2%. Accuracy data (Table I) show DW-FedADMM leading for 5 clients (99.21%) and 15 clients (95.29%) in 10 rounds, while FedADMM leads for 10 clients (95.37%).

For 20 rounds (Table II), DW-FedADMM leads for 15 clients (88.86%) and performs well for 10 clients (93.75%, led by FedADMM at 93.94%) and 5 clients (98.09%, led by FedAvg at 98.40%). Accuracy metrics (Table I) show FedAvg leading for 5 clients (98.93%), FedADMM for 10 clients (97.57%), and DW-FedADMM for 15 clients (95.81%). Accuracy improvements from 10 to 20 rounds (e.g., 94.53% to 96.99% for FedAvg with 10 clients) indicate better convergence, with DW-FedADMM maintaining robust performance despite variability.

Tuandromd Dataset. The Tuandromd dataset shows strong performance with notable effects from client numbers, with DW-FedADMM excelling. In Table II (10 rounds), DW-FedADMM achieves the highest F1 scores for 10 clients (97.86%) and 15 clients (96.76%), while FedADMM leads for 5 clients (98.83%). Specificity is high (e.g., 99.95% for FedProx with 15 clients), but F1 scores drop for 15 clients (e.g., 91.34% for FedProx). Accuracy data (Table I) show FedADMM leading for 5 clients (98.09%) and 10 clients (96.38%) in 10 rounds, while DW-FedADMM leads for 15 clients (94.34%).

For 20 rounds (Table II), DW-FedADMM dominates with the highest F1 scores: 99.13% (5 clients), 98.08% (10 clients), and 97.75% (15 clients). Accuracy metrics (Table I) show DW-FedADMM leading for 5 clients (98.59%) and 10 clients (96.75%, closely following FedADMM at 96.55%), while FedADMM leads for 15 clients (96.29%). Accuracy improvements from 10 to 20 rounds (e.g., 96.75% to 98.59% for DW-FedADMM with 5 clients) align with F1 score gains, highlighting DW-FedADMM's stability.

Overall Comparison. Across all datasets, DW-FedADMM consistently delivers high performance, as shown in Fig. 3. It achieves the highest overall mean accuracy on Malgenome (97.02%) and Tuandromd (96.63%) and performs strongly on Drebin (97.89%, closely following FedADMM at 97.87%). FedADMM also shows robust performance, particularly on Tuandromd (96.60%) and Malgenome (96.39%), while FedAvg and FedProx are competitive but lower (e.g., 95.13% and 95.06% on Malgenome, respectively). DW-FedAvg and DW-FedProx lag slightly, with mean accuracies of 94.68% and 94.53% on Malgenome. Malgenome exhibits high variability, particularly for FedAvg, FedProx, DW-FedAvg, and DW-FedProx, whereas Drebin remains stable. DW-FedADMM's ability to maintain high accuracy across diverse datasets, especially in heterogeneous settings like Malgenome, underscores its robustness in FL scenarios.

D. Algorithm Performance Comparison

DW-FedADMM excels in both F1 and accuracy across datasets, as shown in Fig. 3, with the highest mean accuracy on Malgenome (97.02%) and Tuandromd (96.63%) and strong performance on Drebin (97.89%). For example, on Drebin (20 rounds, 5 clients, Table II), DW-FedADMM achieves an F1 score of 97.69%, matching DW-FedAvg, with accuracy at 98.30% (Table I, led by DW-FedAvg). FedAvg and FedProx show similar performance, with FedProx slightly underperforming due to regularization (e.g., accuracy of 98.28% vs. 98.20% on Drebin). On Malgenome (20 rounds, 15 clients), DW-FedADMM leads with an F1 score of 88.86% and accuracy of 95.81%, despite high variability, outperforming DW-FedAvg (93.31% accuracy) and FedADMM. On Tuandromd, DW-FedADMM dominates with an F1 score of 99.13% (5 clients, 20 rounds) and a mean accuracy of 96.63%, showcasing its robustness.

E. Impact of Client Numbers

Increasing client numbers reduces accuracy, specificity, and F1 scores due to data heterogeneity, but DW-FedADMM performs robustly, as evidenced by its high mean accuracy (Fig. 3: 97.89% on Drebin, 97.02% on Malgenome, 96.63% on Tuandromd). On Malgenome (10 rounds, Table II), DW-FedADMM achieves the highest F1 scores of 88.06% (15 clients) and 88.71% (10 clients), though with high variability, compared to DW-FedAvg's drop from 98.52% (5 clients) to 67.27% (15 clients). On Tuandromd (20 rounds), DW-FedADMM maintains a strong F1 score of 97.75% (15 clients)

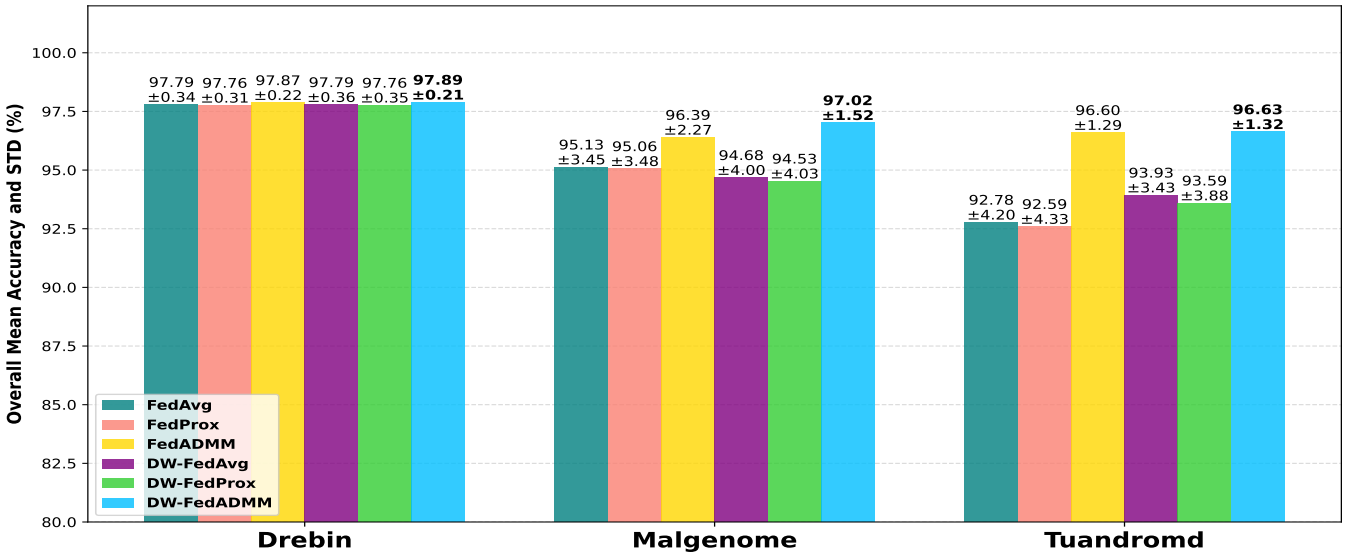

Fig. 3: Overall Mean Accuracy comparison between FL algorithms for each dataset

and accuracy of 98.59% (5 clients), while FedADMM leads at 96.29% accuracy for 15 clients. AUC remains robust (above 99.2%) across datasets, with DW-FedADMM consistently competitive, particularly on Malgenome and Tuandromd.

F. Effect of Training Rounds

Extending from 10 to 20 rounds improves accuracy and F1 scores, with DW-FedADMM showing notable gains (Fig. 3: 97.89% on Drebin, 97.02% on Malgenome, 96.63% on Tuandromd). On Drebin, DW-FedADMM's F1 score for 15 clients increases from 96.77% to 96.78% (Table II), with accuracy for 5 clients improving from 98.16% to 98.30% (Table I). On Malgenome, DW-FedADMM's F1 score rises from 88.06% to 88.86% (15 clients), while FedADMM's F1 improves from 88.16% to 93.94% (10 clients). On Tuandromd, DW-FedADMM excels, with F1 scores increasing from 98.58% to 99.13% (5 clients) and accuracy from 97.64% to 98.59% (5 clients). High variability persists in Malgenome for larger client numbers, but DW-FedADMM's consistent improvements highlight its robustness across datasets.

IV. Conclusion

In this work, we proposed DW-FedADMM, a novel FL framework that unified primal-dual optimization with performance-adaptive aggregation to address the intertwined challenges of heterogeneity, client drift, and partial participation in mobile threat detection. By embedding an ADMM-based primal-dual formulation within each client's local training routine, the algorithm maintained consensus through dual variable coordination, while simultaneously enabling decoupled model updates under communication constraints. The utilization of dynamic weighting mechanisms, derived from client-side performance feedback, further introduced a form of statistical adaptivity to the aggregation process, allowing the global model to evolve based on both optimization geometry and empirical task utility. This dual-informed, weight-adjusted interaction induced a structured regularization effect on the optimization trajectory, mitigating the instability commonly observed in federated non-convex settings. Empirical evaluations on multiple Android malware classification benchmarks revealed that DW-FedADMM not only accelerated convergence but also reduced variance across rounds, achieving superior accuracy, AUC, and robustness compared to state-of-the-art baselines. These findings validated the efficacy of coordinated dual variable dynamics and adaptive client prioritization in advancing scalable, secure, and optimization-efficient federated learning systems.

References

[1] M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, "Dl-droid: Deep learning based android malware detection using real devices," *Computers & Security*, vol. 89, p. 101663, 2020.

[2] Z. Çıplak, K. Yıldız, and Ş. Altınkaya, "Fedetect: A federated learning-based malware detection and classification using deep neural network algorithms," *Arabian Journal for Science and Engineering*, pp. 1–28, 2025.

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

[4] S. Wang, R. Morabito, S. Hosseinalipour, M. Chiang, and C. G. Brinton, "Device sampling and resource optimization for federated learning in cooperative edge networks," *IEEE/ACM Transactions on Networking*, 2024.

[5] D. Thakur, A. Guzzo, and G. Fortino, "Hardware-algorithm co-design of energy efficient federated learning in quantized neural network," *Internet of Things*, vol. 26, p. 101223, 2024.

[6] N. Kumari and P. K. Jana, "Communication efficient federated learning with data offloading in fog-based iot environment," *Future Generation Computer Systems*, vol. 158, pp. 158–166, 2024.

[7] H. Wang and J. Xu, "Friends to help: Saving federated learning from client dropout," in *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 8896–8900.

[8] V. Rey, P. M. S. Sánchez, A. H. Celdrán, and G. Bovet, "Federated learning for malware detection in iot devices," *Computer Networks*, vol. 204, p. 108693, 2022.

TABLE II: Global Models Results for Specificity, AUC, and F1 Across Datasets and Algorithms for 10 and 20 Rounds (in %)

		No. of	FedAvg			FedProx			FedADMM			DW-FedAvg			DW-FedProx			DW-FedADMM		
		Clients	**Spec.**	**AUC**	**F1**	**Spec.**	**AUC**	**F1**	**Spec.**	**AUC**	**F1**	**Spec.**	**AUC**	**F1**	**Spec.**	**AUC**	**F1**	**Spec.**	**AUC**	**F1**
10 Rounds	Drebin	5	**97.29 ±0.27**	99.62 ±0.03	97.56 ±0.27	97.12 ±0.20	99.61 ±0.03	97.47 ±0.21	97.36 ±0.89	99.56 ±0.05	97.38 ±0.42	97.43 ±0.27	**99.63 ±0.03**	**97.58 ±0.27**	97.22 ±0.18	99.62 ±0.03	97.51 ±0.21	97.44 ±0.89	99.59 ±0.05	97.51 ±0.39
		10	96.08 ±0.41	99.54 ±0.06	96.77 ±0.43	96.04 ±0.43	99.54 ±0.06	96.76 ±0.44	**97.46 ±0.80**	**99.58 ±0.10**	**97.29 ±0.70**	96.38 ±0.40	99.53 ±0.08	96.78 ±0.56	96.29 ±0.38	99.53 ±0.09	96.75 ±0.58	97.29 ±0.79	99.56 ±0.12	97.15 ±0.72
		15	95.53 ±1.67	99.43 ±0.23	96.42 ±1.13	95.46 ±1.73	99.43 ±0.24	96.40 ±1.16	**96.70 ±1.56**	99.48 ±0.22	96.65 ±1.28	94.93 ±1.97	99.44 ±0.21	96.26 ±1.22	94.81 ±2.23	99.44 ±0.22	96.23 ±1.26	96.63 ±1.68	**99.50 ±0.25**	**96.77 ±1.38**
	Malgenome	5	95.60 ±7.44	99.94 ±0.09	97.26 ±4.22	95.44 ±7.77	99.94 ±0.09	97.16 ±4.44	**99.07 ±1.12**	99.95 ±0.05	98.45 ±0.65	97.84 ±1.61	99.95 ±0.06	98.52 ±0.81	97.64 ±2.31	99.95 ±0.06	98.39 ±1.16	98.92 ±1.83	**99.96 ±0.07**	**98.83 ±0.87**
		10	84.44 ±29.62	99.73 ±0.53	86.62 ±29.39	84.02 ±29.79	99.73 ±0.55	86.33 ±29.43	87.95 ±29.60	99.76 ±0.53	88.16 ±29.45	76.83 ±38.57	99.58 ±0.84	78.12 ±39.10	76.76 ±38.57	99.58 ±0.81	78.07 ±39.09	**88.96 ±29.67**	**99.80 ±0.41**	**88.71 ±29.57**
		15	67.88 ±39.38	99.37 ±1.00	71.42 ±39.10	67.53 ±39.38	99.36 ±1.02	71.14 ±39.14	79.23 ±39.62	99.65 ±0.80	78.72 ±39.36	65.10 ±42.73	99.25 ±1.35	67.27 ±43.78	64.25 ±42.45	99.24 ±1.34	66.69 ±43.77	**88.11 ±29.59**	**99.85 ±0.22**	**88.06 ±29.40**
	Tuandromd	5	98.39 ±0.64	99.76 ±0.19	97.84 ±2.68	98.41 ±0.64	99.76 ±0.19	97.84 ±2.68	98.82 ±0.74	**99.79 ±0.17**	98.83 ±0.74	**98.90 ±0.56**	99.78 ±0.16	98.10 ±2.48	98.98 ±0.53	99.78 ±0.17	98.09 ±2.52	98.83 ±0.71	**99.77 ±0.23**	98.58 ±1.44
		10	98.82 ±0.76	99.44 ±0.43	95.90 ±3.68	98.82 ±0.76	99.44 ±0.43	95.74 ±3.80	98.43 ±0.97	**99.65 ±0.34**	97.89 ±2.72	**98.91 ±0.68**	99.45 ±0.53	96.33 ±3.36	99.09 ±0.74	99.42 ±0.60	96.20 ±3.37	98.53 ±1.02	**99.68 ±0.28**	97.86 ±2.76
		15	**99.93 ±0.21**	98.41 ±3.25	91.54 ±2.84	99.95 ±0.16	98.39 ±3.28	91.34 ±2.68	98.52 ±1.56	99.20 ±1.27	96.69 ±3.53	99.74 ±0.56	98.75 ±2.07	92.73 ±3.66	99.93 ±0.21	98.72 ±2.16	92.00 ±3.38	98.59 ±1.57	**98.96 ±1.74**	**96.76 ±3.55**
20 Rounds	Drebin	5	**97.33 ± 0.22**	99.63 ± 0.03	97.67 ± 0.23	97.14 ± 0.18	99.62 ± 0.03	97.55 ± 0.19	97.25 ± 0.51	99.55 ± 0.06	97.28 ± 0.46	97.41 ± 0.20	**99.64 ± 0.03**	**97.69 ± 0.22**	97.30 ± 0.18	99.63 ± 0.03	97.63 ± 0.19	97.24 ± 0.59	99.57 ± 0.05	97.35 ± 0.46
		10	96.02 ± 0.52	99.57 ± 0.06	96.95 ± 0.36	95.94 ± 0.53	99.56 ± 0.06	96.91 ± 0.38	**97.31 ± 0.65**	99.56 ± 0.07	**97.34 ± 0.59**	96.05 ± 0.58	99.57 ± 0.06	96.93 ± 0.41	96.03 ± 0.35	99.55 ± 0.07	96.94 ± 0.35	97.21 ± 0.85	**99.58 ± 0.10**	97.28 ± 0.67
		15	95.62 ± 1.13	99.50 ± 0.17	96.55 ± 0.92	95.59 ± 1.19	99.49 ± 0.18	96.53 ± 0.92	**96.99 ± 1.49**	**99.54 ± 0.19**	96.77 ± 1.45	95.44 ± 1.42	99.48 ± 0.18	96.60 ± 0.90	95.41 ± 1.40	99.49 ± 0.19	96.57 ± 0.87	96.94 ± 1.84	99.53 ± 0.16	**96.78 ± 1.42**
	Malgenome	5	97.51 ± 2.95	**99.96 ± 0.05**	**98.40 ± 1.53**	97.37 ± 3.00	99.96 ± 0.05	98.34 ± 1.56	98.30 ± 2.51	99.95 ± 0.06	97.92 ± 2.05	96.74 ± 5.15	99.96 ± 0.06	98.01 ± 2.91	96.41 ± 7.34	99.96 ± 0.05	97.71 ± 4.43	**98.84 ± 1.28**	99.94 ± 0.06	98.09 ± 2.13
		10	91.62 ± 21.51	99.83 ± 0.38	93.00 ± 21.49	91.58 ± 21.51	99.83 ± 0.39	92.98 ± 21.48	**94.31 ± 21.64**	**99.93 ± 0.15**	**93.94 ± 21.56**	92.07 ± 21.25	99.88 ± 0.22	93.29 ± 21.44	91.45 ± 21.10	99.88 ± 0.21	92.97 ± 21.36	94.23 ± 21.63	99.91 ± 0.23	93.75 ± 21.52
		15	80.52 ± 34.19	99.72 ± 0.40	82.38 ± 34.71	80.37 ± 34.19	99.71 ± 0.40	82.27 ± 34.69	89.21 ± 29.75	**99.85 ± 0.38**	88.60 ± 29.54	80.98 ± 34.31	99.69 ± 0.61	82.58 ± 34.77	80.46 ± 34.11	99.69 ± 0.65	82.31 ± 34.67	**89.38 ± 29.80**	99.82 ± 0.51	**88.86 ± 29.63**
	Tuandromd	5	99.00 ± 0.89	**99.81 ± 0.18**	98.63 ± 1.88	99.00 ± 0.89	99.80 ± 0.18	98.59 ± 1.99	**99.15 ± 0.75**	99.80 ± 0.21	98.83 ± 1.38	98.96 ± 0.48	99.78 ± 0.25	98.51 ± 2.02	98.97 ± 0.51	99.78 ± 0.25	98.49 ± 2.02	99.05 ± 0.58	99.86 ± 0.08	**99.13 ± 0.29**
		10	98.65 ± 0.74	99.52 ± 0.76	96.66 ± 3.44	98.66 ± 0.76	99.51 ± 0.79	96.63 ± 3.43	**98.80 ± 1.03**	99.67 ± 0.42	97.99 ± 2.75	98.49 ± 0.75	99.47 ± 1.00	97.11 ± 3.11	98.53 ± 0.74	99.49 ± 0.93	97.10 ± 3.11	98.68 ± 0.97	**99.71 ± 0.30**	**98.08 ± 2.19**
		15	**99.09 ± 0.90**	99.25 ± 1.15	94.89 ± 4.06	99.11 ± 0.93	99.24 ± 1.16	94.73 ± 4.07	98.65 ± 1.23	99.50 ± 0.73	**97.84 ± 2.71**	98.74 ± 0.81	99.44 ± 0.89	96.44 ± 3.39	98.84 ± 0.82	99.45 ± 0.84	96.26 ± 3.52	98.80 ± 1.24	**99.55 ± 0.77**	97.75 ± 2.74

[9] W. Fang, J. He, W. Li, X. Lan, Y. Chen, T. Li, J. Huang, and L. Zhang, "Comprehensive android malware detection based on federated learning architecture," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 3977–3990, 2023.

[10] M. E. Khoda, T. Imam, J. Kamruzzaman, I. Gondal, and A. Rahman, "Robust malware defense in industrial iot applications using machine learning with selective adversarial samples," *IEEE Transactions on Industry Applications*, vol. 56, no. 4, pp. 4415–4424, 2019.

[11] R. Taheri, M. Shojafar, F. Arabikhan, and A. Gegov, "Unveiling vulnerabilities in deep learning-based malware detection: Differential privacy driven adversarial attacks," *Computers & Security*, vol. 146, p. 104035, 2024.

[12] I. Gulrajani, C. Raffel, and L. Metz, "Towards gan benchmarks which require generalization," *arXiv preprint arXiv:2001.03653*, 2020.

[13] W. Huang, M. Ye, Z. Shi, G. Wan, H. Li, B. Du, and Q. Yang, "Federated learning for generalization, robustness, fairness: A survey and benchmark," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[14] L. Lyu, H. Yu, X. Ma, C. Chen, L. Sun, J. Zhao, Q. Yang, and S. Y. Philip, "Privacy and robustness in federated learning: Attacks and defenses," *IEEE transactions on neural networks and learning systems*, 2022.

[15] R. Taheri, M. Shojafar, M. Alazab, and R. Tafazolli, "Fed-iiot: A robust federated malware detection architecture in industrial iot," *IEEE transactions on industrial informatics*, vol. 17, no. 12, pp. 8442–8452, 2020.

[16] A. Khraisat, A. Alazab, S. Singh, T. Jan, and A. Jr. Gomez, "Survey on federated learning for intrusion detection system: Concept, architectures, aggregation strategies, challenges, and future directions," *ACM Computing Surveys*, vol. 57, no. 1, pp. 1–38, 2024.

[17] L. Li, W. Xu, T. Chen, G. B. Giannakis, and Q. Ling, "Rsa: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 1544–1551.

[18] A. Acharya, A. Hashemi, P. Jain, S. Sanghavi, I. S. Dhillon, and U. Topcu, "Robust training in high dimensions via block coordinate geometric median descent," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2022, pp. 11 145–11 168.

[19] C. Jiang, K. Yin, C. Xia, and W. Huang, "Fedhgcdroid: An adaptive multi-dimensional federated learning for privacy-preserving android malware classification," *Entropy*, vol. 24, no. 7, p. 919, 2022.

[20] D. Hamouda, M. A. Ferrag, N. Benhamida, Z. E. Kouahla, and H. Seridi, "Android malware detection based on network analysis and federated learning," in *Cyber Malware: Offensive and Defensive Systems*. Springer, 2023, pp. 23–39.

[21] B. Li, Y. Wu, J. Song, R. Lu, T. Li, and L. Zhao, "Deepfed: Federated deep learning for intrusion detection in industrial cyber–physical systems," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5615–5624, 2021.

[22] H. Yang, L. Cheng, and M. C. Chuah, "Deep-learning-based network intrusion detection for scada systems," in *2019 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2019, pp. 1–7.

[23] D. Javeed, M. S. Saeed, M. Adil, P. Kumar, and A. Jolfaei, "A federated learning-based zero trust intrusion detection system for internet of things," *Ad Hoc Networks*, vol. 162, p. 103540, 2024.

[24] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[25] S. Zhu, J. Zeng, S. Wang, Y. Sun, X. Li, Y. Yao, and Z. Peng, "On admm in heterogeneous federated learning: Personalization, robustness, and fairness," *arXiv preprint arXiv:2407.16397*, 2024.

[26] H. Wang, S. Marella, and J. Anderson, "Fedadmm: A federated primal-dual algorithm allowing partial participation," in *2022 IEEE 61st Conference on Decision and Control (CDC)*. IEEE, 2022, pp. 287–294.

[27] A. Chaudhuri, A. Nandi, and B. Pradhan, "A dynamic weighted federated learning for android malware classification," in *Soft computing: Theories and applications: Proceedings of SoCTA 2022*. Springer, 2023, pp. 147–159.

[28] S. Y. Yerima and S. Sezer, "Droidfusion: A novel multilevel classifier fusion approach for android malware detection," *IEEE transactions on cybernetics*, vol. 49, no. 2, pp. 453–466, 2018.

[29] Y. Zhang, C. Jiang, B. Yue, J. Wan, and M. Guizani, "Information fusion for edge intelligence: A survey," *Information Fusion*, vol. 81, pp. 171–186, 2022.

[30] P. Borah, D. Bhattacharyya, and J. Kalita, "Malware dataset generation and evaluation," in *2020 IEEE 4th Conference on Information & Communication Technology (CICT)*. IEEE, 2020, pp. 1–6.