applied sciences

MDPI

Article

Client Selection in Federated Learning on Resource-Constrained Devices: A Game Theory Approach

Zohra Dakhia [1,2,†] **and Massimo Merenda** [2,3,*,†]

[1] Department of Biology, University Federico II of Naples, 80126 Naples, Italy; zohra.dakhia@unirc.it
[2] Department of Information Engineering, Infrastructures and Sustainable Energy, University Mediterranea of Reggio Calabria, 89124 Reggio Calabria, Italy
[3] HWA srl, Spin-Off Mediterranea University of Reggio Calabria, Via R. Campi II tr. 135, 89126 Reggio Calabria, Italy
* Correspondence: massimo.merenda@unirc.it
† These authors contributed equally to this work.

Abstract

Federated Learning (FL), a key paradigm in privacy-preserving and distributed machine learning (ML), enables collaborative model training across decentralized data sources without requiring raw data exchange. FL enables collaborative model training across decentralized data sources while preserving privacy. However, selecting appropriate clients remains a major challenge, especially in heterogeneous environments with diverse battery levels, privacy needs, and learning capacities. In this work, a centralized reward-based payoff strategy (RBPS) with cooperative intent is proposed for client selection. In RBPS, each client evaluates participation based on locally measured battery level, privacy requirement, and the model's accuracy in the current round computing a payoff from these factors and electing to participate if the payoff exceeds a predefined threshold. Participating clients then receive the updated global model. By jointly optimizing model accuracy, privacy preservation, and battery-level constraints, RBPS realizes a multi-objective selection mechanism. Under realistic simulations of client heterogeneity, RBPS yields more robust and efficient training compared to existing methods, confirming its suitability for deployment in resource-constrained FL settings. Experimental analysis demonstrates that RBPS offers significant advantages over state-of-the-art (SOA) client selection methods, particularly those relying on a single selection criterion such as accuracy, battery, or privacy alone. These one-dimensional approaches often lead to trade-offs where improvements in one aspect come at the cost of another. In contrast, RBPS leverages client heterogeneity not as a limitation, but as a strategic asset to maintain and balance all critical characteristics simultaneously. Rather than optimizing performance for a single device type or constraint, RBPS benefits from the diversity of heterogeneous clients, enabling improved accuracy, energy preservation, and privacy protection all at once. This is achieved by dynamically adapting the selection strategy to the strengths of different client profiles. Unlike homogeneous environments, where only one capability tends to dominate, RBPS ensures that no key property is sacrificed. RBPS thus aligns more closely with real-world FL deployments, where mixed-device participation is common and balanced optimization is essential.

Keywords: federated learning; client selection; game theory; cooperative learning; heterogeneous devices; reward-based payoff strategy

Academic Editors: David Sarabia-Jácome and Carlos Enrique Palau Salvador

Received: 14 June 2025
Revised: 2 July 2025
Accepted: 3 July 2025
Published: 5 July 2025

Citation: Dakhia, Z.; Merenda, M. Client Selection in Federated Learning on Resource-Constrained Devices: A Game Theory Approach. *Appl. Sci.* **2025**, *15*, 7556. https://doi.org/10.3390/app15137556

1. Introduction

Federated Learning (FL) has emerged as a transformative paradigm for collaborative machine learning (ML), enabling a network of distributed clients such as smartphones, edge devices, and IoT nodes to train a shared global model without sharing their raw data [1]. This decentralized approach offers substantial privacy advantages but introduces new challenges in managing system resources and client diversity. A core challenge in FL is the efficient selection of participating clients at each communication round, as this decision directly affects model convergence, battery consumption, and privacy leakage [2]. In practical deployments, FL systems often consist of highly heterogeneous devices: resource-rich smartphones, latency-sensitive edge devices, and battery-constrained IoT sensors. These devices vary not only in computational power and battery capacity but also in privacy sensitivity and data quality. As a result, client selection becomes a multi-objective decision-making problem, where a trade-off must be struck between model accuracy, battery sustainability, and privacy preservation. The multi-objective client selection problem in FL requires not only rigorous theoretical modeling but also careful consideration of real-world system constraints. These include managing energy consumption on battery-powered devices, preserving user privacy, and adapting to the heterogeneity of modern IoT and edge computing environments. By combining game-theoretic concepts with these practical challenges, our approach delivers a scalable and adaptive solution that bridges theoretical foundations with deployment realities. This integration ensures that FL can be effectively implemented in diverse, resource-constrained settings, moving beyond purely conceptual frameworks toward impactful, operational systems. Moreover, our strategy accounts for realistic client diversity and operational variability, fostering solutions that are both theoretically sound and practically viable. This comprehensive perspective enables the design of mechanisms that meet multiple objectives simultaneously (accuracy, sustainability, and privacy) reflecting the complex demands of modern distributed learning applications. Prior works typically optimize for a single objective, for example selecting clients solely based on data quality [3] or battery availability [4,5], and therefore fail to adapt to the complex, multi-objective nature of real-world FL scenarios. To address this limitation, we propose a centralized reward-based payoff strategy (RBPS) with cooperative intent for client selection. RBPS formulates the selection process as a cooperative game, where clients collectively align with a system-wide objective by optimizing a shared reward function. Using the principles of Nash Equilibrium (NE), RBPS balances three critical metrics: model accuracy, battery level, and privacy sensitivity. Each client evaluates a reward function that integrates these factors, with adaptive weights assigned according to its device type (smartphone, edge, or IoT). Unlike prior approaches that treat clients as passive participants or optimize one dimension at a time, RBPS is implemented as a centralized strategy with cooperative intent, dynamically adapting to client contexts. In each round, clients with favorable reward scores are selected to participate and receive the updated model, while non-selected clients remain idle to preserve battery life and privacy budgets. Our RBPS strategy is designed to reflect real-world deployment constraints. We simulate a heterogeneous pool of devices with varying levels of battery, computation, and privacy settings, and evaluate the strategy using four benchmark datasets of increasing complexity: MNIST, EMNIST, Fashion-MNIST, and CIFAR-10. Clients are drawn from realistic distributions, including smartphones (high accuracy, moderate privacy, and high battery consumption), edge devices (balanced capabilities), and IoT sensors (low battery consumption and high privacy risk).

The key contributions of this paper are as follows:

1. A multi-objective client selection strategy: we propose RBPS, a game-theoretic method based on NE, that jointly considers accuracy, battery level, and privacy sensitivity. The method dynamically adjusts each client's reward using contextual device information.
2. Cooperative participation logic: clients with the highest rewards are selected to contribute and receive updated models. Non-selected clients remain inactive, reducing unnecessary battery usage and limiting the exposure of sensitive data.
3. Realistic simulation of heterogeneous devices: we simulate a broad range of clients, including smartphones, edge nodes, and IoT devices, with assigned profiles reflecting real-world battery, privacy, and accuracy characteristics.
4. Comprehensive empirical evaluation: we perform extensive experiments using four benchmark datasets (MNIST, EMNIST, Fashion-MNIST, and CIFAR-10) over 300 communication rounds. Our evaluation spans various client pool sizes (10, 25, 50, 100, 1000, and 5000) and includes both homogeneous client configurations (e.g., all IoT or all smartphones) and heterogeneous client pools that mix devices of different capabilities. We also compare RBPS against several state-of-the-art (SOA) methods, including Oort, FedCS, FRS, and FedProm, to assess its effectiveness in balancing accuracy, battery preservation, and privacy.
5. Analysis of trade-offs: our results highlight that RBPS provides a better balance across accuracy, battery sustainability, and privacy preservation, particularly in large, heterogeneous client pools where other methods either sacrifice performance or overburden specific device classes.

By addressing all three core FL objectives in a unified and adaptive framework, RBPS offers a scalable, fair, and sustainable solution for FL in real-world, battery-constrained environments. The remainder of this paper is organized as follows. Section 2 reviews existing works related to client selection in FL, highlighting their limitations in handling multi-objective trade-offs. Section 3 introduces the necessary preliminaries and concepts used throughout this paper. In Section 4, we present our game theoretic client selection model based on NE. Section 5 discusses how we realistically model resource constraints such as battery level and privacy. Section 6 explains the experimental setup and evaluation procedure. Section 7 presents the results and discussion, including insights on the impact of client pool sizes and a comparison with existing methods. Finally, Section 8 concludes this paper and outlines directions for future work.

2. Related Work

Client selection plays a central role in FL, as it determines which subset of devices contributes to each training round. While early works relied on random selection, recent research has moved toward intelligent and adaptive selection mechanisms that better reflect real-world constraints such as device heterogeneity, battery limitations, and privacy requirements. One of the first notable attempts to move beyond random selection is federated client selection (FedCS) [6], which selects clients based on their computational and communication capabilities. This method laid the foundation for system-aware FL by acknowledging that not all devices are equally capable. However, it primarily targets deadline compliance and does not adapt to dynamic battery states or consider user privacy. To incorporate data quality and responsiveness into the selection process, Oort [7] introduced a utility-based client selection framework using reinforcement learning. Oort builds on the idea of system-awareness from FedCS but enhances it with real-time data-driven feedback. Nevertheless, it continues to neglect privacy constraints and does not explicitly model battery usage. Recognizing the need for fairness in participation, fair resource scheduling (FRS) [8] shifts the focus toward balanced resource allocation across heterogeneous clients.

FRS takes a step further by promoting equitable use of devices over time. However, like its predecessors, it lacks mechanisms for real-time privacy control or battery-aware selection. To address personalization and user-side priorities, FedProm [9] proposes a preference and fairness aware selection scheme. While it introduces user centric customization, FedProm is still constrained by assumptions of uniform hardware profiles and does not adapt to battery or privacy dynamics during runtime. From a network-efficiency perspective, FedGCS [10] leverages dynamic graphs to optimize latency-aware client selection. It brings in network topology as a selection factor but assumes homogeneous device characteristics and overlooks both battery and privacy concerns. An alternative path is taken by multi device aggregation (MDA) [11], which enables several low-power devices to collaboratively contribute to model updates. This method improves inclusion and participation from battery constrained devices, but it does not enforce any safeguards on battery usage or sensitive data exposure. Another grouping-based method, tiered FL (TiFL) [12], clusters devices into static performance-based tiers to mitigate straggler effects. TiFL improves efficiency in device selection but is rigid and incapable of adapting to changing client contexts like battery status or data sensitivity. Beyond the discussed approaches, several recent studies have advanced client selection by integrating privacy preservation and battery efficiency with more granular adaptation. For instance, Ref. [13] proposes a privacy-preserving client selection scheme leveraging differential privacy to prevent sensitive information leakage during training, while [14] introduces an energy-aware scheduling algorithm that dynamically adjusts client participation to optimize device lifetime. Other works, such as [15], explore personalized FL client selection that considers both user preferences and device heterogeneity, addressing some limitations of static or hardware-homogeneous assumptions. These contributions underscore the multi-faceted nature of client selection challenges, yet they often treat energy, privacy, or personalization in isolation, without a unified framework balancing these critical factors simultaneously. Despite these advances, current methods either target individual aspects (e.g., accuracy, fairness, or latency) or assume homogeneity in client capabilities. They do not fully embrace the multi-objective, dynamic nature of real-world deployments. This reveals a significant knowledge gap: the lack of an adaptive selection strategy that accounts for accuracy contribution, energy constraints, privacy sensitivity, and heterogeneous devices all at once. Our proposed RBPS addresses this gap through a centralized strategy with cooperative intent, tailored to heterogeneous and battery-constrained environments. RBPS evaluates each client through a multi-objective reward function that dynamically balances three core dimensions: expected accuracy contribution, current battery level, and privacy sensitivity. Unlike prior methods, RBPS updates client eligibility at every round, ensuring that only suitable devices are selected. This centralized approach with cooperative intent reduces battery drain, avoids client overuse, and respects user-specific constraints. Furthermore, RBPS benefits from client heterogeneity rather than being hindered by it. While many SOA methods are optimized for uniform environments and thus tend to favor only one capability (e.g., maximizing accuracy at the cost of battery), our strategy uses heterogeneity to balance performance across all critical aspects. This makes RBPS more robust, generalizable, and practically viable for real-world FL systems. Table 1 summarizes this contrast. In the broader context, adaptive client selection strategies like RBPS are essential as FL continues to move toward deployment in edge and IoT scenarios, where device diversity and resource constraints are the norm. Moreover, with increasing interest in federated multi-task learning and privacy regulations, client selection mechanisms must become even more flexible and multi-objective. Our work contributes toward this evolving landscape by providing a concrete, multi-objective framework for practical FL systems.

Table 1. Comparison of client selection strategies across key criteria. ✓ (green): Feature is addressed by the method; ✗ (red): Feature is not addressed by the method.

Method	Accuracy-Aware	Battery-Aware	Privacy-Aware	Heterogeneous Devices	Cooperative	Selective Update
FedCS [6]	✓	✓	✗	✓	✗	✗
Oort [7]	✓	✓	✗	✓	✗	✗
FRS [8]	✓	✓	✗	✓	✓	✗
FedProm [9]	✓	✗	✗	✗	✓	✗
FedGCS [10]	✓	✓	✗	✗	✗	✗
MDA [11]	✓	✗	✗	✓	✓	✗
TiFL [12]	✓	✓	✗	✓	✗	✗
RBPS (Ours)	✓	✓	✓	✓	✓	✓

3. Preliminaries

This section provides a brief overview of the key concepts and models used throughout this paper, including FL, NE, and the client selection framework based on our RBPS. These concepts form the foundation of the proposed solution and are essential to understanding the client selection process in FL.

3.1. Federated Learning

FL is a distributed ML paradigm that allows multiple clients to collaboratively train a shared model while keeping their data decentralized and private. In FL, clients, such as IoT devices, smartphones, or edge devices, each maintain a local dataset and perform local model training on their respective data. Instead of sending raw data to a central server, clients send only model updates (gradients) to the server, which aggregates these updates to improve the global model. This process preserves data privacy and reduces communication costs associated with centralized data processing [16,17].

Mathematically, FL can be formulated as follows:

$$\min_{\theta} \left(\sum_{k=1}^{K} \frac{n_k}{n} F_k(\theta) \right) \tag{1}$$

where

- θ is the global model parameter vector;
- $F_k(\theta)$ is the local objective function (e.g., cross-entropy loss) on client k's data;
- n_k is the number of data points on client k;
- $n = \sum_{k=1}^{K} n_k$ is the total number of data points across all K clients.

This formulation ensures that each client's contribution is weighted proportionally to its data size. It was introduced in the seminal work by McMahan et al. [18] and widely used in the FL literature [16,17].

Federated Learning Algorithms

Several algorithms have been developed to address the core challenges of FL, particularly in handling data distribution, communication efficiency, and system heterogeneity. Two widely used and foundational algorithms are Federated averaging (FedAvg) and FedProx.

- FedAvg is a basic aggregation method proposed by McMahan et al. [18] It works by selecting a subset of clients in each training round. Each selected client performs local model updates on its private data using stochastic gradient descent (SGD) or another optimizer. The clients then send their updated model parameters to a central server, which aggregates them (typically through weighted averaging based on dataset sizes)

to form the new global model. FedAvg is efficient in terms of communication and computation and serves as the standard baseline in many FL studies.

- FedProx proposed by Li et al. [19] extends FedAvg by introducing a proximal term to the local objective function. This regularization term penalizes significant divergence of local model parameters from the global model, which helps stabilize training when client data is non-IID (i.e., not independently and identically distributed). FedProx is particularly useful in heterogeneous environments where clients may have diverse data distributions, computation capacities, and participation frequencies.

These algorithms lay the groundwork for more advanced FL strategies and serve as key benchmarks for evaluating new techniques in federated settings.

3.2. Client Selection in FL

Client selection is crucial in FL as it directly impacts the efficiency, convergence speed, and accuracy of the training process [20]. The clients selected in each training round contribute to model updates, so it is essential to choose clients that balance computational capacity, resource availability, and privacy concerns.

Several client selection strategies have been proposed, each focusing on different criteria, such as the following:

- Random Selection: clients are chosen randomly, often used as a baseline.
- Battery Optimization: prioritizing clients with sufficient battery levels to prevent premature device shutdowns during training.
- Privacy Preservation: choosing clients based on their privacy needs, ensuring that the most privacy-sensitive data is processed by devices with adequate security.
- Accuracy-based Selection: selecting clients with high local model accuracy to speed up convergence.

3.3. Nash Equilibrium

NE is a fundamental concept in game theory (GT), particularly suited for scenarios in which multiple decision makers, here the clients, interact in a shared environment and their outcomes are interdependent [21,22]. In traditional non-cooperative games, each player aims to maximize its own payoff, assuming the strategies of others are fixed. However, in RBPS, the client selection process is modeled as a cooperative game, where clients are indirectly aligned through a shared reward function that integrates three system-wide objectives: model accuracy, battery efficiency, and privacy preservation. Each client evaluates its participation based on local conditions such as remaining battery level, data sensitivity, and expected accuracy contribution and responds accordingly. While the NE condition still applies, it is interpreted in a cooperative sense: no client has an incentive to deviate unilaterally from the collectively optimized strategy, as doing so would reduce its reward relative to the joint outcome.

Mathematically, the NE is expressed as follows:

$$u_i(s_i^*, s_{-i}^*) \geq u_i(s_i, s_{-i}^*) \quad \forall s_i \in S_i \tag{2}$$

where u_i denotes the reward (payoff) function for client i, s_i^* is the equilibrium strategy of client i, and s_{-i}^* represents the strategy profile of all other clients. At equilibrium, no client benefits by changing its selection strategy alone.

In our framework, the payoff function u_i is computed using a weighted combination of the client's impact on accuracy, its battery consumption profile, and privacy sensitivity. These weights are adjusted dynamically based on the client's device type and real-time context. RBPS then centrally selects clients with cooperative intent, ensuring that participa-

tion decisions lead to an overall system-optimal outcome rather than allowing any single dimension (e.g., accuracy or fairness) to dominate the selection logic.

3.4. Reward-Based Payoff Strategy

The RBPS is designed to address the limitations of traditional client selection strategies by incorporating a combination of accuracy, battery level, and privacy into a single, unified framework. In RBPS, the payoff for each client is determined by a weighted sum of the following components:

- Accuracy: the local model's accuracy on the client's data.
- Battery Level: the remaining battery percentage of the device.
- Privacy: the privacy level of the data, which is impacted by the client's ability to maintain confidentiality.

The overall payoff function for client i in the RBPS framework is defined as follows:

$$\text{Payoff}_i = \omega_A \cdot \text{A}_i + \omega_B \cdot \text{B}_i + \omega_P \cdot \text{P}_i \tag{3}$$

where

- A_i is the contribution to model accuracy (e.g., historical validation performance);
- B_i is the normalized battery level;
- P_i is the privacy preference or sensitivity score (e.g., set by user or device type);
- ω_A, ω_B, and ω_P are weights that adjust the importance of each factor based on the client's characteristics.

The weights ω_A, ω_B, and ω_P are dynamically adjusted based on the current state of each client. This allows the strategy to better reflect the practical constraints and priorities of devices during training. For example, if a client's battery level drops significantly, the system can increase ω_B to reduce the likelihood of selecting that device, helping to preserve its energy. If a client contributes consistently to model improvement, ω_A may be increased to prioritize its selection. Similarly, when stronger privacy preferences are detected, ω_P can be raised to give more consideration to privacy concerns. This dynamic adjustment makes the selection process more adaptive and context-aware. Rather than relying on fixed priorities, the system responds to real-time information from each client, promoting a balanced trade-off between model accuracy, energy efficiency, and privacy protection.

3.5. Battery Consumption and Privacy Loss

The battery consumption of clients and the privacy loss due to client selection are key factors in the RBPS framework. The battery consumption decreases when clients are selected for training because their device's battery is used for computation. Conversely, when a client is not selected, it still consumes battery to receive updates from the server but at a reduced rate [23,24].

We model battery consumption as follows:

3.5.1. Case 1: Device Participates in Training

When a device participates in FL training, its battery decreases based on workload and device efficiency. The battery at the next time step, bat_{t+1}, is given by the following equation:

$$bat_{t+1} = bat_t - \alpha \cdot bs \cdot C_m \cdot W_d \cdot \text{steps per epoch} \tag{4}$$

where

- bs: batch size;

- C_m: model complexity;
- steps per epoch: number of steps per epoch;
- W_d: data size weight;
- α: device-specific energy cost constant (based on mAh per operation).

This formula captures the compute load and energy cost per training step, in line with edge-AI energy modeling [25,26].

3.5.2. Case 2: Device Does Not Participate in Training

If a device does not participate in training, the battery decreases only due to background activities or idle battery consumption. This can be modeled as follows:

$$bat_{t+1} = bat_t - \beta \tag{5}$$

where β is a fixed idle battery drain constant, estimated based on typical standby energy consumption observed in edge and mobile devices [27].

Similarly, privacy loss is modeled based on how much information can be inferred from the client's participation in FL. The weights and biases of a neural network (NN) can contain indirect information about the dataset used to train the model. While these parameters do not directly store the raw data, they are learned from it during the training process. As the model optimizes its weights and biases to minimize the loss function, these parameters reflect patterns and features specific to the training data. For example, in Model Inversion Attacks, an adversary can exploit these learned weights to reconstruct features or even specific samples from the training dataset, despite never directly accessing the data itself [28]. Additionally, in Membership Inference Attacks, attackers can infer whether a particular data point was part of the training set by analyzing the model's predictions and the confidence levels of those predictions [29]. This is possible because a model's behavior typically differs for data seen during training and unseen data. For instance, the model may provide more confident predictions or overfit to specific patterns learned during training, allowing attackers to identify membership. These privacy vulnerabilities demonstrate that while FL minimizes direct data sharing, the model's parameters themselves may still reveal sensitive information about individual client data, necessitating the use of privacy-preserving techniques such as differential privacy [18]. Over time, the cumulative exposure of sensitive information due to repeated participation in FL results in privacy degradation, reflecting a gradual decline in the client's data confidentiality. Each interaction causes a quantifiable privacy loss, representing the specific amount of information potentially leaked during that training round. Moreover, the degree to which a client's data or device is vulnerable to such privacy risks depends on its privacy sensitivity—an intrinsic characteristic influenced by factors such as data uniqueness, device security, and context. The impact on privacy as a client participates in the FL process can be modeled as follows:

$$priv_{t+1} = priv_t - \delta \cdot \frac{\|\mathcal{D}_{update}\|}{\text{total number of clients}} \tag{6}$$

where

- $priv_t$ represents the privacy level at time t;
- $priv_{t+1}$ is the privacy level after the client has participated in the next round;
- δ is the privacy degradation factor, a parameter that defines how much each update reduces privacy;
- $\mathcal{D}_{update}$ refers to the magnitude of the update sent by the client to the server, which is determined by the difference in the client's local model compared to the global model;
- The total number of clients is included to account for the dilution of privacy as more clients contribute updates to the global model.

This formula represents how the privacy level of a client diminishes over time as they participate in the FL process and share updates with the server. As more clients contribute to the model, the privacy loss is diluted, but it is still a key consideration in the selection process [30,31].

4. Game Theoretic Client Selection Model

This section introduces our RBPS as a centralized strategy with cooperative intent, framing it within the context of a cooperative game where clients contribute to a shared objective by participating in the FL training process based on a common reward-based scoring strategy. By applying the concept of NE, we identify stable selection outcomes in which no client can improve its reward by deviating alone. Figure 1 shows the main structure of our system, where RBPS is used to determine which clients should join the FL process. In this setup, each client is evaluated based on its current battery level, data sensitivity, and potential accuracy contribution. The server selects the top-scoring clients to participate, collects their updates, and sends back the improved model. Clients that are not selected do not participate and do not receive the model, helping conserve battery and protect privacy. We use NE to reach a cooperative and stable situation, where no client has an incentive to change its state unilaterally, and where selection benefits both individual clients and the global model performance.

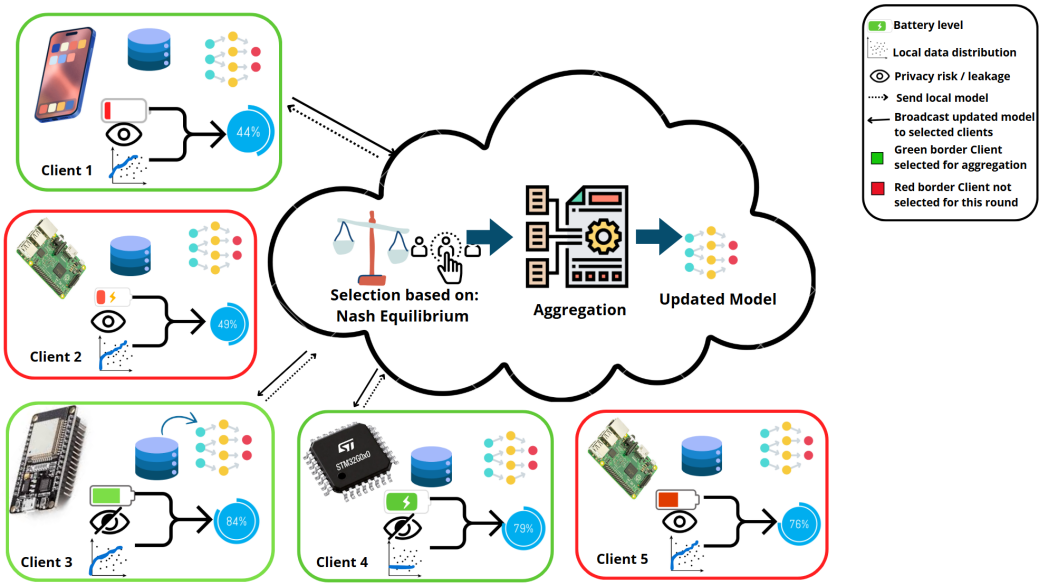

Figure 1. System architecture of the proposed RBPS using NE in FL environments.

4.1. System Setup and Objective

Let $\mathcal{C} = \{1, 2, \ldots, N\}$ denote the set of clients in a FL system. In each communication round t, every client $i \in \mathcal{C}$ chooses an action:

- $a_i^t = 1$: participate in training.
- $a_i^t = 0$: skip training.

Each client is characterized by the following parameters:

- $A_i^t \in [0, 1]$: current local model accuracy.
- $B_i^t \in [0, 1]$: normalized battery level.
- $P_i^t \in [0, 1]$: privacy preference (higher values indicate stronger preferences).

The global objective is to select a subset $\mathcal{S}_t \subset \mathcal{C}$ such that the overall system utility is maximized while considering accuracy, battery sustainability, and privacy preservation. As each client acts selfishly, a game-theoretic formulation is essential.

4.2. Nash Game Formulation

Each client i is a player in the game defined by:

- Strategy set: $a_i^t \in \{0, 1\}$
- Utility function:

$$U_i(a_i^t, a_{-i}^t) = a_i^t \cdot (\omega_A A_i^t + \omega_B B_i^t + \omega_P P_i^t) - a_i^t \cdot (\lambda_B \Delta B_i^t + \lambda_P \Delta P_i^t) \tag{7}$$

where

- a_{-i}^t represents the action profile of all clients other than i.
- ω_A, ω_B, and $\omega_P \in [0, 1]$ are time-varying reward weights that satisfy the following condition:

$$\omega_A + \omega_B + \omega_P = 1. \tag{8}$$

  These weights are dynamically updated at each training round based on the real-time state of each client. They reflect three key aspects: the client's contribution to model accuracy (ω_A), battery constraints (ω_B), and privacy sensitivity (ω_P). The adjustment of these weights is grounded in both theoretical reasoning Formulas (4)–(6) and empirical observations. In practice, involving a client in training improves global model accuracy, but at the cost of energy consumption and potential privacy risks. Therefore, to balance this trade-off, raw utility scores are computed from current client metrics: recent local model accuracy, normalized battery level, and a predefined privacy score. These scores are then normalized to ensure the resulting weights sum to one. This dynamic weighting allows the system to prioritize clients that provide high utility while considering their limitations. For example, clients with low battery or high privacy sensitivity receive higher ω_B or ω_P, which lowers their chance of being selected. In contrast, clients contributing significantly to accuracy are favored through increased ω_A. This adaptive mechanism ensures fairness, improves training efficiency, and allows the strategy to perform well under diverse and changing edge environments.
- λ_B, λ_P are penalty weights for battery and privacy degradation, respectively.
- ΔB_i^t and ΔP_i^t denote the expected battery drop and privacy loss.

  Each client aims to maximize their utility:

$$a_i^{t*} = \arg \max_{a_i^t \in \{0,1\}} U_i(a_i^t, a_{-i}^t) \tag{9}$$

  A NE is achieved when the following conditions are met:

$$U_i(a_i^{t*}, a_{-i}^{t*}) \geq U_i(a_i^t, a_{-i}^{t*}) \quad \forall a_i^t \in \{0, 1\}, \forall i \tag{10}$$

4.3. Client Selection Algorithm

As shown in Algorithm 1, at each round, all clients decide simultaneously. Participation is based on whether the reward exceeds a threshold.

Algorithm 1 Nash-Based Client Participation Decision

Require: Accuracy A_i, Battery B_i, Privacy P_i for each client i; threshold τ
Ensure: Selected set of clients $\mathcal{S}_t$
for each client $i \in \mathcal{C}$ **do**
  Assign $\omega_A, \omega_B, \omega_P$ based on device type
  $R_i \leftarrow \omega_A A_i + \omega_B B_i + \omega_P P_i$ reward (payoff) from accuracy, battery, privacy
  if $R_i \geq \tau$ **then**
    $a_i \leftarrow 1$ participate
  else
    $a_i \leftarrow 0$ skip
  end if
end for
$\mathcal{S}_t \leftarrow \{ i \in \mathcal{C} \mid a_i = 1 \}$
return $\mathcal{S}_t$

4.4. Battery and Privacy Cost Estimation

While exact values for battery drop and privacy degradation vary across devices, our model assumes that each client estimates the following:

- ΔB_i^t: battery consumption expected if selected for training at round t.
- ΔP_i^t: privacy degradation expected if selected for training at round t.

These cost estimations depend on the device's capabilities and sensitivity. For example, lightweight IoT sensors typically incur lower battery costs but may suffer greater privacy risks due to limited data protection mechanisms.

5. Realistic Modeling of Resource Constraints

To ensure the viability and applicability of our game-theoretic framework for FL, we integrate realistic resource constraints directly into the client decision-making process. Our system captures heterogeneity in hardware capabilities, battery availability, communication bandwidth, and privacy sensitivity. These parameters influence each client's utility function and are critical in shaping participation strategies across repeated FL rounds. The battery and privacy parameters used in this study were derived from real-world device specifications and prior empirical studies to ensure that the simulation reflects realistic operating conditions. Battery capacities and power consumption profiles were obtained from manufacturer datasheets and hardware performance reports for devices such as the ESP32 microcontroller [25], the Jetson Nano embedded platform [32], and common android smartphones like the Google Pixel 6 [33]. The battery consumption per training round was aligned with values reported in recent edge AI benchmarks and FL performance analyses [26]. Privacy sensitivity levels for IoT and edge devices were modeled based on foundational work in privacy attacks on FL systems [28,29], which highlight the higher vulnerability of low-end devices to inference risks. These references collectively support the realism and credibility of the simulation model adopted in RBPS.

5.1. Device Classification and Capabilities

We categorize participating clients into three classes based on their hardware profiles and operational environments, as shown in Table 2.

Table 2. Device classes and characteristics.

Device Type	Examples	CPU	Battery (mAh)	Uplink	Privacy Risk	Notes
IoT Devices [34]	ESP32, Arduino	Low	200–500	Slow	High	Battery constrained and privacy-sensitive
Edge Devices [35]	Raspberry Pi, Jetson Nano	Medium	2000–4000	Moderate	Medium	Moderate performance, battery-aware
Smartphones [36,37]	Pixel, Galaxy, etc.	High	3000–5000	Fast	Low	High performance, latency-sensitive

These characteristics influence the parameters of each client's utility function, guiding their strategic decisions to participate or not in each FL round.

5.2. Battery Consumption Modeling

Battery cost is one of the principal deterrents for participation in FL. The estimated battery usage per device type is summarized in Table 3. For each client i, we model the relative battery drop as follows:

$$\Delta B_i = \begin{cases} \frac{B^i_{\text{sel}}}{B^i_{\text{total}}} & \text{if selected for training and communication} \\ \frac{B^i_{\text{idle}}}{B^i_{\text{total}}} & \text{otherwise} \end{cases}$$

where B^i_{sel} is the battery consumed when participating in a round, B^i_{idle} is the idle battery consumption, and B^i_{total} denotes the current battery level.

Table 3. Battery usage estimates per 100 rounds.

Device Type	B^i_{sel} (mAh)	B^i_{idle} (mAh)	Avg. Drop (%)
IoT Devices	5–15	<1	3–4%
Edge Devices	50–100	5–10	5–7%
Smartphones	150–250	20–30	8–10%

This battery model is embedded into the client's payoff function through a cost term $\lambda_B \cdot \Delta B_i$, allowing the system to naturally favor battery-efficient participation patterns.

5.3. Privacy Loss Modeling

Client participation in FL can result in varying degrees of privacy exposure depending on the nature of their local data. We quantify the privacy cost of client i as follows:

$$\Delta P_i = \alpha_i \cdot f(D_i) \tag{11}$$

where $\alpha_i \in [0, 1]$ is the intrinsic privacy sensitivity of device i, and $f(D_i)$ captures the sensitivity of the data being used for local training. The privacy sensitivity coefficients α_i for different device types are summarized in Table 4.

Table 4. Privacy sensitivity by device type.

Device Type	α_i
IoT Devices	0.8–1.0
Edge Devices	0.4–0.6
Smartphones	0.2–0.5

This cost term, $\lambda_P \cdot \Delta P_i$, is integrated into the reward function, discouraging frequent selection of highly privacy-sensitive clients unless they provide critical contributions.

6. Experimental Evaluation

This section presents an extensive evaluation of our proposed NE-based client selection strategy, comparing it against baseline and SOA methods across different datasets, client pool sizes, and device profiles. Our evaluation emphasizes model accuracy, fairness, battery efficiency, and privacy preservation.

6.1. Experimental Setup

6.1.1. Datasets

We evaluate our approach using several image classification datasets that vary in complexity. For each dataset, different model architectures are used to reflect the characteristics of the dataset and optimize the performance. The choice of MNIST, EMNIST, Fashion-MNIST, and CIFAR-10 is deliberate, as each presents distinct challenges that are valuable for assessing the robustness and scalability of our proposed RBPS method. MNIST provides a simple, well-established benchmark for basic image recognition tasks, making it ideal for testing the foundational aspects of our model. EMNIST introduces the complexity of handwritten letters, expanding the task to include both digits and characters, which requires a more nuanced approach. Fashion-MNIST further increases the challenge with more abstract patterns and item categorization, demanding higher-level feature extraction capabilities. CIFAR-10, with its diverse set of color images across 10 different classes, offers a significantly more complex task, requiring advanced models capable of handling a wide variety of objects and features. By selecting these datasets, we ensure that our method is tested across a range of tasks that vary in difficulty and data distribution. This progression from simpler to more complex datasets allows us to evaluate the adaptability of our approach in real-world scenarios, where data complexity can vary widely. Additionally, the use of these standard datasets aligns with FL benchmarks, providing a consistent basis for comparison with existing methods in the field. Testing our approach on these datasets also enables us to assess how well our client selection strategy, based on GT, adapts to diverse data distributions, which is crucial for ensuring the efficiency, scalability, and privacy of our solution in federated settings.

- MNIST: the MNIST dataset consists of 60,000 grayscale images of handwritten digits (0–9) for training and 10,000 images for testing. Each image is 28 × 28 pixels.
- EMNIST: the EMNIST dataset is an extension of MNIST, including both digits and letters (uppercase and lowercase). It contains 814,255 characters for training and 141,292 characters for testing.
- Fashion-MNIST: Fashion-MNIST contains 60,000 28 × 28 grayscale images of 10 fashion categories, with 10,000 test images. This dataset is a more challenging alternative to MNIST.
- CIFAR-10: The CIFAR-10 dataset consists of 60,000 32 × 32 color images categorized into 10 different classes, with 6000 images per class.

6.1.2. Model Architecture

Each dataset required specific model configurations to achieve optimal performance. Below, we outline the architectures used for each dataset.

- For MNIST and Fashion-MNIST: a simpler convolutional neural network (CNN) architecture was used:
  - Input layer: 28 × 28 pixels for MNIST, 32 × 32 for Fashion-MNIST.
  - Convolutional layers:
    - First convolutional layer with 32 filters of size 3 × 3 and ReLU activation;
    - Second convolutional layer with 64 filters of size 3 × 3 and ReLU activation.

  - Fully connected layer: dense layer with 128 units and ReLU activation.
  - Output layer: 10 units (corresponding to 10 classes) with softmax activation.
- For EMNIST: given the increase in data complexity and the inclusion of both digits and letters, a slightly deeper model with additional convolutional layers was used:
  - Input Layer: 28 × 28 pixels.
  - Convolutional layers:
    - First convolutional layer with 32 filters of size 3 × 3 and ReLU activation;
    - Second convolutional layer with 64 filters of size 3 × 3 and ReLU activation;
    - Third convolutional layer with 128 filters of size 3 × 3 and ReLU activation.
  - Fully connected layer: dense layer with 256 units and ReLU activation.
  - Output layer: 62 units (corresponding to 62 classes) with softmax activation.
- For CIFAR-10: a deeper CNN architecture was used to handle the complexity of the CIFAR-10 dataset:
  - Input Layer: 32 × 32 pixels.
  - Convolutional layers:
    - First convolutional layer with 32 filters of size 3 × 3 and ReLU activation;
    - Second convolutional layer with 64 filters of size 3 × 3 and ReLU activation;
    - Third convolutional layer with 128 filters of size 3 × 3 and ReLU activation.
  - Fully connected layer: dense layer with 512 units and ReLU activation.
  - Output layer: 10 units (corresponding to 10 classes) with softmax activation.

6.2. Model Parameters

The following Table 5 summarizes the key parameters used in the CNN model for each dataset.

Table 5. Key model parameters used across different datasets.

Parameter	MNIST & Fashion-MNIST	EMNIST	CIFAR-10
Optimizer	Adam	Adam	Adam
Learning Rate	0.001	0.001	0.001
Batch Size	32	64	64
Epochs	20	30	50
Convolution layer 1	32 filters, 3 × 3, ReLU	32 filters, 3 × 3, ReLU	32 filters, 3 × 3, ReLU
Convolution layer 2	64 filters, 3 × 3, ReLU	64 filters, 3 × 3, ReLU	64 filters, 3 × 3, ReLU
Convolution layer 3	—	128 filters, 3 × 3, ReLU	128 filters, 3 × 3, ReLU
Fully connected layer	128 units, ReLU	256 units, ReLU	512 units, ReLU
Output layer	10 units (softmax)	62 units (softmax)	10 units (softmax)
Loss function	Categorical crossentropy	Categorical crossentropy	Categorical crossentropy

These parameters were optimized for each dataset, ensuring that the models could effectively learn from the data while maintaining efficient training times and computational cost.

6.3. Simulation of Device Heterogeneity

To reflect realistic deployment scenarios, we simulated a heterogeneous mix of devices, including IoT devices, edge devices, and smartphones. Each device was modeled with varying levels of battery capacity and privacy sensitivity, based on typical characteristics observed in real-world applications. The simulation was conducted using Google Co-

lab, which provides virtual machines equipped with NVIDIA Tesla T4 GPUs, Intel Xeon CPUs, and approximately 12.7 GB of RAM. The FL experiments employed Flower (Flwr) version 1.19.0, an open-source framework for FL. The environment ran Python 3.8 with TensorFlow 2.12 for model training and evaluation. Additional libraries such as NumPy 1.23.5 and Matplotlib 3.7.1 were used for data processing and visualization. This setup enabled efficient simulation of client–server interactions and dynamic training rounds under realistic conditions. In each simulation run, and for every tested client pool size (e.g., 10, 25, 50, etc.), we ensured that the selected clients were a mix of different device types. This heterogeneous grouping was meant to reflect real-world scenarios where various devices with different capabilities participate in training. The mixture of devices in each pool allowed us to evaluate how battery levels, privacy sensitivity, and contribution to accuracy influence participation and overall training performance. The participation behavior of clients determined by their profile directly impacted the global model's accuracy, average battery consumption, and privacy loss.

7. Results and Discussion

7.1. Impact of Device Heterogeneity on FL Performance

To evaluate how client diversity impacts federated performance, we designed a controlled comparison using 100 clients across three configurations:

- Homogeneous pools: all clients of a single type (smartphones, edge, or IoT).
- Semi-heterogeneous pools: 50 clients from each of two types (e.g., smartphones + edge).
- Fully heterogeneous pools: 33 smartphones, 33 edge devices, and 34 IoT devices.

Client characteristics (battery, privacy sensitivity, and accuracy potential) followed the profiles in Table 2. Experiments were conducted using four datasets of increasing difficulty and visual complexity, ranging from grayscale handwritten digits to color image classification. This gradient in data complexity allows us to observe how device heterogeneity affects performance as the representational and computational demands of the task increase.

7.1.1. Impact on Accuracy Across Datasets

Across all datasets, as shown in Figures 2–5, fully heterogeneous (3-type) pools consistently delivered superior and more stable accuracy compared to homogeneous (1-type) and semi-heterogeneous (2-type) pools.

- Figure 2: while all pools performed well, homogeneous smartphone-only pools reached 91% accuracy more quickly. However, the fully heterogeneous 3-type pool achieved a similar result while maintaining a smoother convergence path due to the diverse contributions from different devices.
- Figure 3: the 2-type and 3-type heterogeneous pools outperformed the homogeneous 1-type pools. IoT-only and edge-only pools struggled to reach competitive accuracy, with the 3-type pool maintaining steady gains through the mid and late rounds.
- Figure 4: the performance gap widened, with homogeneous 1-type pools overfitting quickly or slowing down, particularly the smartphone-only pools. In contrast, the 3-type heterogeneous pools preserved accuracy and reached 88% with less fluctuation.
- Figure 5: this more complex task, requiring deeper models and better generalization, showed that homogeneous 1-type pools plateaued below 75%. The 3-type heterogeneous pools, however, steadily climbed to 78%, highlighting the importance of device diversity as model complexity increases.

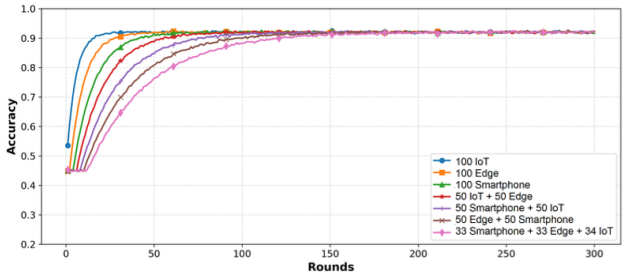

Figure 2. Impact of device heterogeneity on accuracy over 300 rounds (MNIST).

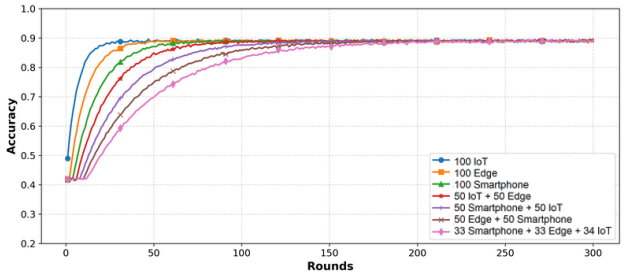

Figure 3. Impact of device heterogeneity on accuracy over 300 rounds (EMNIST).

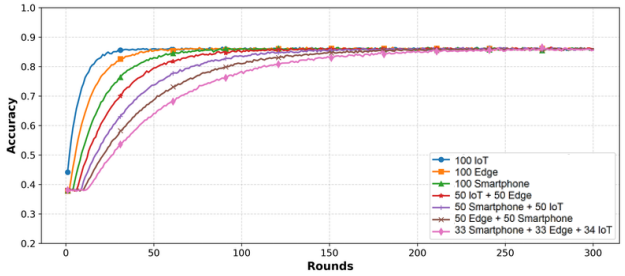

Figure 4. Impact of device heterogeneity on accuracy over 300 rounds (Fashion-MNIST).

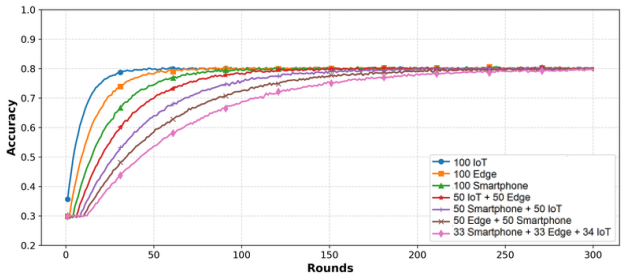

Figure 5. Impact of device heterogeneity on accuracy over 300 rounds (CIFAR-10).

As the datasets become more abstract and challenging, the advantage of including multiple device types becomes increasingly evident. Heterogeneous participation leads to better generalization, helping to avoid bias toward any one class of local data.

7.1.2. Impact on Battery Consumption Across Datasets

- Figure 6: homogeneous smartphone-only pools experienced sharp battery drops (>9% per round), while IoT-only pools consumed less battery (3%) but made minimal progress. In contrast, the fully heterogeneous 3-type pools balanced battery usage effectively: smartphones contributed early, while edge and IoT devices helped later, ensuring more sustainable battery consumption.
- Figures 7–9: as the models grew in size, the battery burden increased. Homogeneous 1-type pools, particularly smartphone-only ones, became unsustainable. On the other

hand, the 3-type heterogeneous pools intelligently distributed the workload across devices, preventing early dropout and maintaining stable total battery usage.

Only the fully heterogeneous 3-type setup ensured sustainable training without exhausting any particular device group. The diversity of devices allowed for adaptive battery management, which is crucial for real-world deployments with long training cycles.

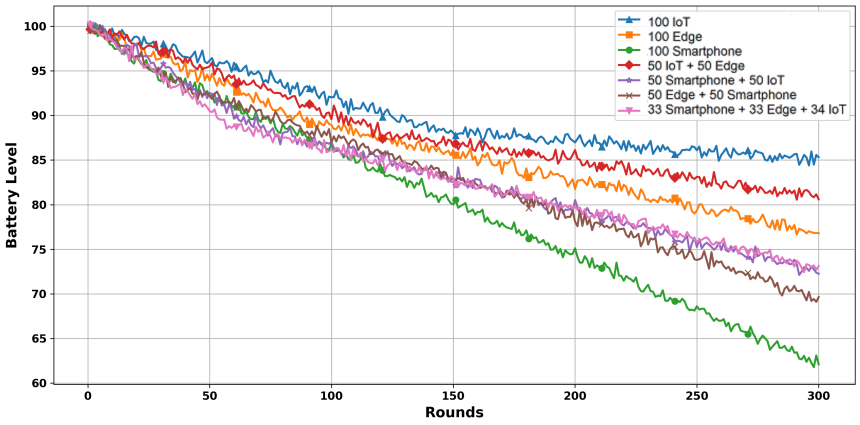

Figure 6. Impact of device heterogeneity on battery over 300 rounds (MNIST).

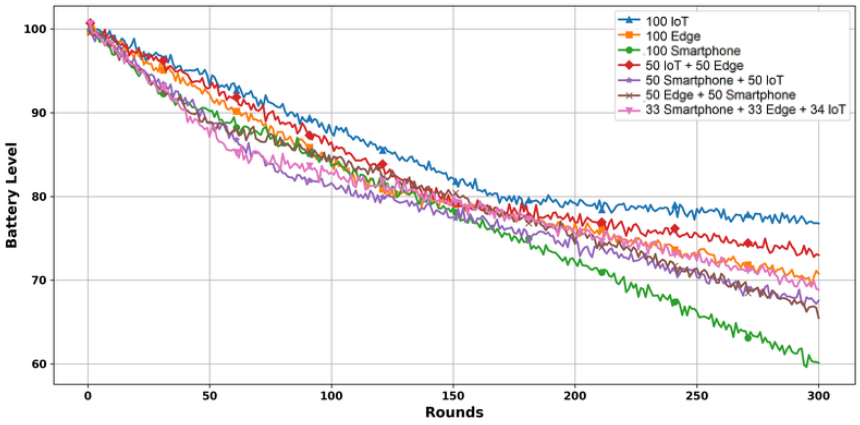

Figure 7. Impact of device heterogeneity on battery over 300 rounds (EMNIST).

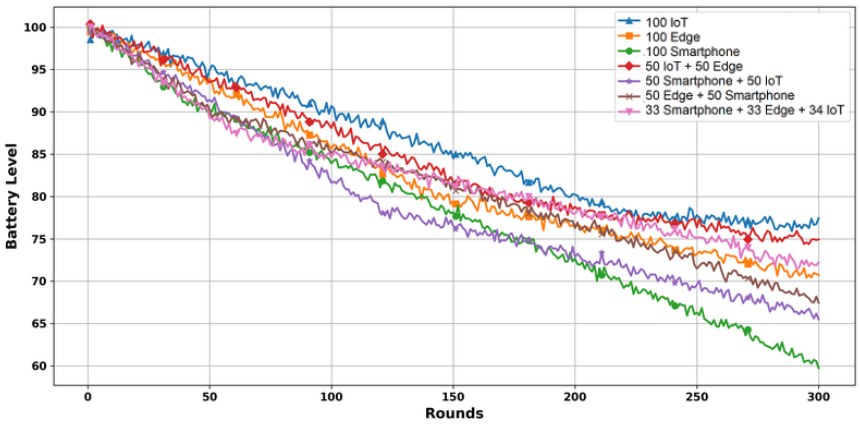

Figure 8. Impact of device heterogeneity on battery over 300 rounds (Fashion-MNIST).

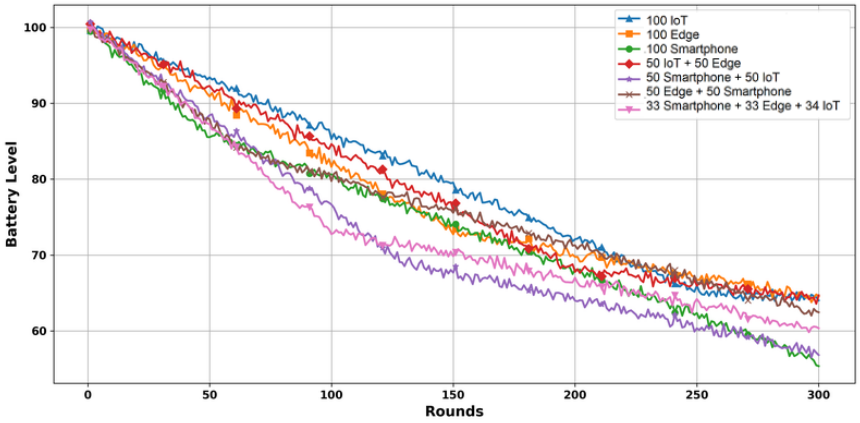

Figure 9. Impact of device heterogeneity on battery over 300 rounds (CIFAR-10).

7.1.3. Impact on Privacy Across Datasets

As illustrated in Figures 10–13, the privacy preservation behavior of FL systems is highly dependent on the type of participating devices. Across all datasets, it is evident that the smartphone-only configuration maintains the highest privacy scores throughout the training process. This confirms that smartphones, which are simulated with higher privacy sensitivity, are either selected less frequently or incur lower privacy loss per round, making them the most robust in terms of privacy protection. Conversely, IoT-only clients experience the most rapid privacy degradation. These devices are modeled with low privacy sensitivity, leading to frequent selection and quick exhaustion of their privacy budgets, especially noticeable in the EMNIST and CIFAR-10 datasets, where the score falls below 0.85 within the first 100 rounds. Mixed-device configurations, such as (edge + smartphone) and (IoT + smartphone), exhibit moderate and more stable privacy trends. The All Devices setting reflects a balanced selection pattern, benefiting from the presence of higher-privacy smartphones while still engaging less-privacy-sensitive IoT nodes. This validates the potential of heterogeneous client pools to achieve privacy-aware training without relying solely on any single device category. Furthermore, we observe that dataset complexity impacts the rate of privacy decline. In simpler datasets like MNIST and Fashion-MNIST, privacy scores stabilize earlier, whereas more complex datasets like EMNIST and CIFAR-10 require more communication and client involvement, resulting in greater cumulative privacy loss. These results confirm that device heterogeneity plays a crucial role in privacy dynamics, and that strategic client selection especially with RBPS can preserve privacy more effectively than uniform or homogenous selection strategies.

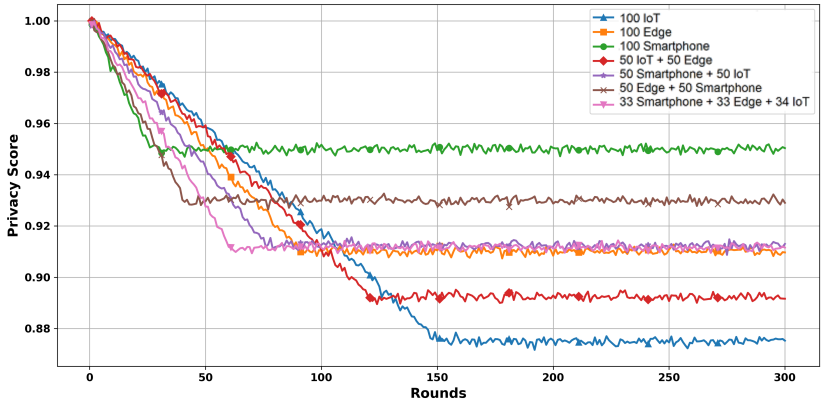

Figure 10. Impact of device heterogeneity on privacy over 300 rounds (MNIST).

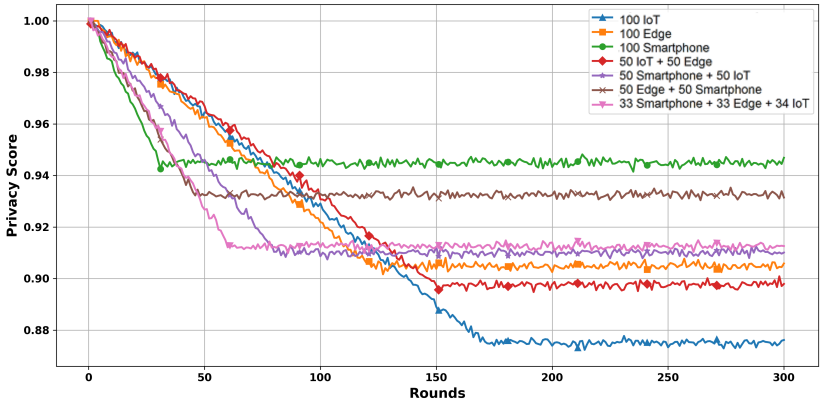

Figure 11. Impact of device heterogeneity on privacy over 300 rounds (EMNIST).

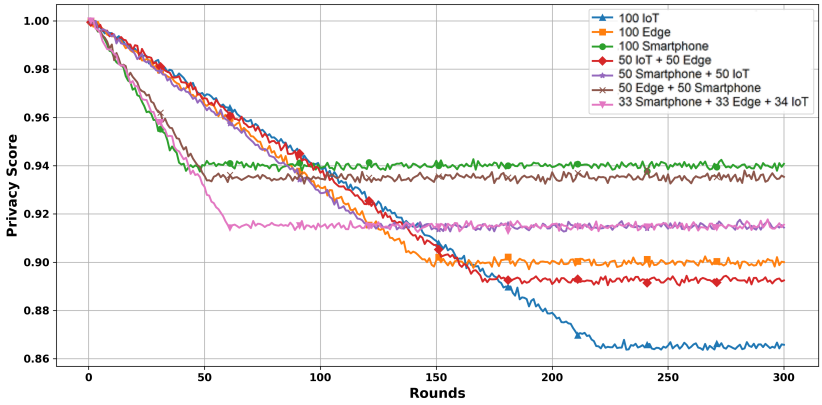

Figure 12. Impact of device heterogeneity on privacy over 300 rounds (Fashion-MNIST).

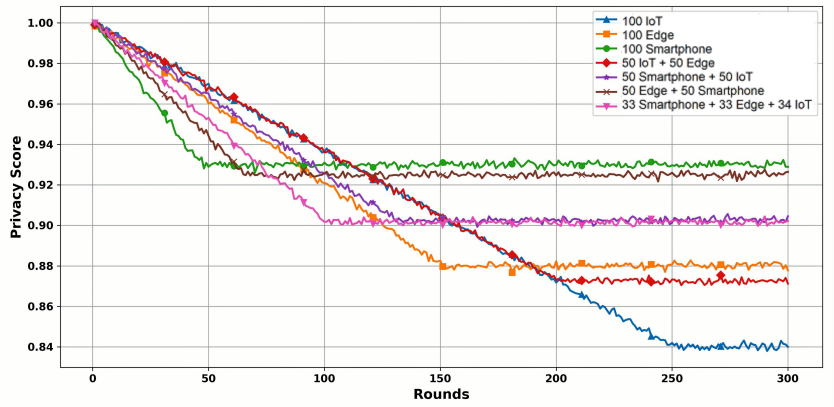

Figure 13. Impact of device heterogeneity on privacy over 300 rounds (CIFAR-10).

7.1.4. Effect of Heterogeneous Client Distribution on FL Trade-Offs

To gain a deeper understanding of how the distribution of client types affects FL performance, we analyze different configurations by adjusting the proportions of IoT, edge, and smartphone devices within the client pool. Each configuration represents a unique allocation strategy, from highly heterogeneous setups (e.g., 60% IoT, 20% edge, and 20% smartphone) to more balanced combinations (e.g., 33% of each device type). We then evaluate the performance of each setup using three key metrics: accuracy, battery consumption, and privacy loss. The results of this comprehensive analysis are summarized in Figure 14.

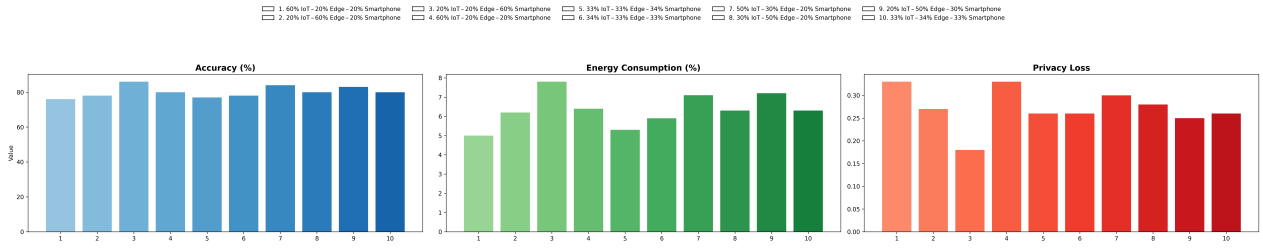

Figure 14. Accuracy, energy consumption, and privacy loss across different heterogeneous client distributions (100 clients; 300 rounds).

Figure 14 clearly shows that client composition has a significant effect on the overall trade-off between accuracy, battery usage, and privacy preservation. Configurations dominated by smartphones (e.g., 60% smartphone) achieve the highest accuracy and the lowest privacy loss but also incur higher battery consumption due to their higher computation demand. In contrast, IoT-heavy configurations consume less battery but perform poorly in accuracy and privacy metrics. The most balanced results are observed in heterogeneous distributions, particularly the setup with approximately 33% of each client type, which yields stable accuracy while moderately preserving battery and privacy. These results reinforce the value of maintaining device diversity in federated training, and they highlight the effectiveness of RBPS in navigating trade-offs in heterogeneous environments.

7.2. Impact of Client Pool Size on Accuracy, Battery, and Privacy

To assess how the size of the client pool influences FL performance under real-world conditions, we examined training behavior on four datasets (MNIST, EMNIST, Fashion-MNIST, and CIFAR-10) using heterogeneous client groups only. In all configurations, the total number of clients remained 100% heterogeneous, with device proportions preserved across:

- 10 clients: approx. 3 smartphones, 3 edge, and 4 IoT;
- 100 clients: 33 smartphones, 33 edge, and 34 IoT;
- 1000 clients: 330 smartphones, 330 edge, and 340 IoT.

All results reported below are measured at round 90, to ensure consistency across settings. Table 6 summarizes the results.

Table 6. Performance summary at round 90 for heterogeneous client pools (adjusted for realistic battery usage).

Dataset	Clients	Accuracy (%)	Battery Usage (%)	Avg. Privacy Loss
MNIST	10	94	18.0	0.40
	100	89	12.0	0.25
	1000	64	7.0	0.12
EMNIST	10	90	19.0	0.44
	100	85	13.0	0.27
	1000	60	8.0	0.13
Fashion-MNIST	10	91	17.0	0.38
	100	86	11.0	0.22
	1000	61	6.5	0.11
CIFAR-10	10	80	20.0	0.47
	100	75	14.0	0.29
	1000	50	9.0	0.14

As shown in Table 6 and Figure 15, increasing the total number of heterogeneous clients leads to a decrease in model accuracy at a fixed round, such as round 90. For

instance, MNIST accuracy drops from approximately 94% with 10 clients to 89% with 100 and 64% with 1000. This trend holds across EMNIST, Fashion-MNIST, and CIFAR-10. The accuracy reduction is primarily due to the lower probability of each individual client being selected in larger pools. As fewer updates are aggregated per round, the global model receives less information, slowing convergence. Figure 16 highlights a benefit of this reduced selection frequency: lower battery consumption per device. In small pools, devices such as smartphones are selected frequently and experience high battery drain (up to 10–11% per round). In contrast, in larger pools, individual clients are chosen less often, reducing average battery usage to around 4–5% and allowing devices to participate longer. Figure 17 shows a similar effect in terms of privacy. In smaller pools, high-sensitivity devices are exposed repeatedly, leading to rapid privacy degradation. Larger pools distribute participation across a wider population, decreasing the likelihood of repeated selection and reducing cumulative privacy loss. For example, in CIFAR-10, the average privacy loss per smartphone falls from 0.47 with 10 clients to just 0.14 with 1000. This analysis demonstrates that while larger client pools may reduce accuracy at a fixed round, they significantly improve battery efficiency and privacy protection by lowering per-device selection probability. These trade-offs highlight the importance of balancing pool size with training duration when designing scalable FL systems.

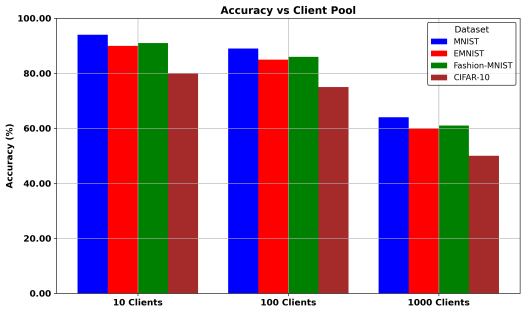

Figure 15. Effect of client pool size on model accuracy.

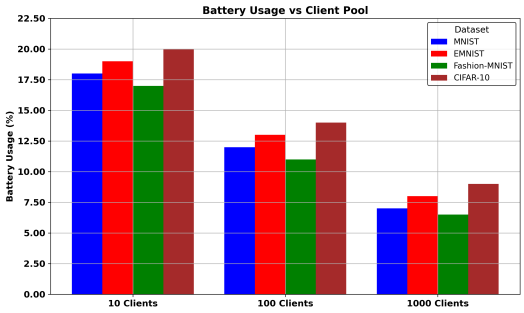

Figure 16. Effect of client pool size on battery usage (%).

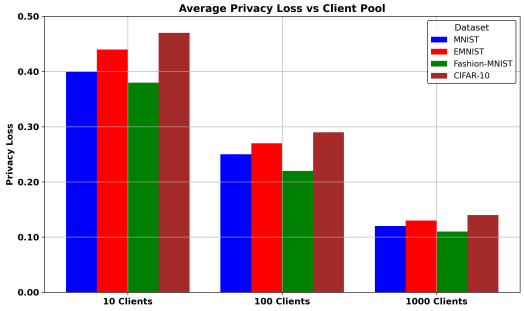

Figure 17. Effect of client pool size on privacy loss.

7.3. Comparative Analysis of Client Selection Strategies

We evaluated the effectiveness of different client selection strategies using a pool of 100 heterogeneous clients (33 smartphones, 33 edge devices, and 34 IoT). All strategies were tested on the same four datasets (MNIST, EMNIST, Fashion-MNIST, and CIFAR-10) over 300 training rounds. The performance was compared across three dimensions: model accuracy, battery consumption, and privacy degradation.

The following strategies were examined:

- Random selection: clients are selected uniformly at random;
- Accuracy-only selection: chooses clients based on their past contribution to model accuracy;
- Battery-only selection: prioritizes clients with the highest remaining battery power;
- Privacy-only selection: selects clients based on minimizing privacy risks;
- RBPS.

7.3.1. Accuracy Performance over 300 Rounds

The accuracy-only strategy achieved the highest performance in the initial training rounds, particularly on MNIST Figure 18 and EMNIST Figure 19, due to its tendency to consistently select clients with the best local model accuracy, typically smartphones. However, by ignoring battery status and privacy sensitivity, it overuses a small subset of devices, which leads to battery depletion and privacy exhaustion over time. As a result, some high-performing clients may become unavailable in later rounds, either due to simulated battery loss or privacy constraints. This causes the accuracy curve to flatten prematurely, especially in more demanding datasets like Fashion-MNIST Figure 20 and CIFAR-10 Figure 21, where generalization requires broader device participation. Random selection performed inconsistently, with modest accuracy in MNIST but weak convergence in EMNIST and CIFAR-10 due to the absence of any utility-driven prioritization. Battery-only and privacy-only strategies underperformed across all datasets, as they selected clients with minimal training cost or risk but often low contribution to learning progress. RBPS delivered competitive accuracy in simple datasets (e.g., 91% on MNIST; 87% on EMNIST) and outperformed all other methods on Fashion-MNIST (88%) and CIFAR-10 (80%), demonstrating its ability to sustain training quality by balancing accuracy potential with battery and privacy constraints throughout the learning process.

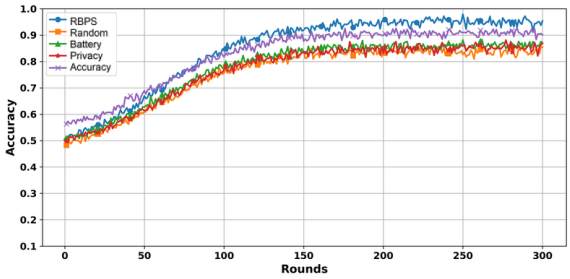

Figure 18. Accuracy over 300 rounds (MNIST).

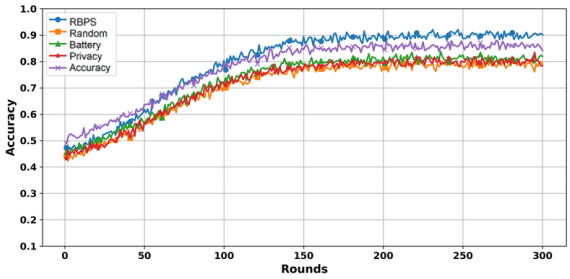

Figure 19. Accuracy over 300 rounds (EMNIST).

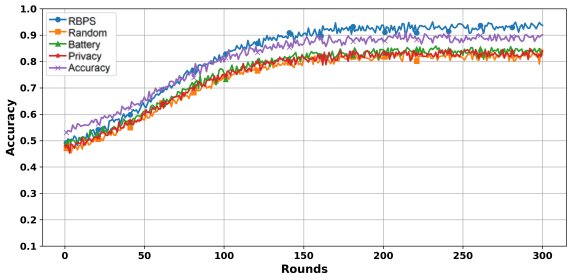

Figure 20. Accuracy over 300 rounds (Fashion-MNIST).

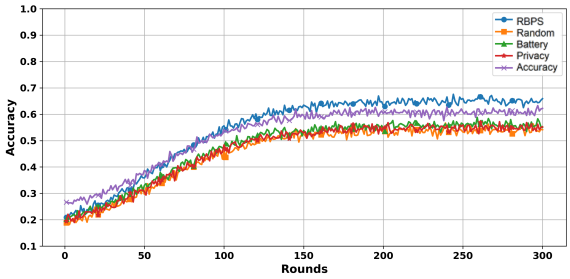

Figure 21. Accuracy over 300 rounds (CIFAR-10).

7.3.2. Loss Reduction over 300 Rounds

The training loss curves across all datasets further illustrate the trade-offs among selection strategies. The accuracy-only strategy demonstrated the fastest initial loss reduction, especially in MNIST Figure 22 and EMNIST Figure 23, due to its repeated selection of clients with strong local models. However, this advantage diminished over time as the same high-performing clients were repeatedly selected without regard for battery or privacy. As these clients began to drop out or contribute less due to constraints, the model's ability to continue improving stalled, and loss reduction slowed significantly in later rounds. Random selection exhibited unstable and fluctuating loss behavior, particularly on EMNIST and CIFAR-10, due to the inconsistency in client quality and data contribution. Battery-only and privacy-only strategies maintained high loss throughout training, reflecting their limited capacity to select impactful clients. These approaches often neglected clients with high gradient contribution, resulting in slower and less effective learning. RBPS, on the other hand, achieved the most stable and consistent loss reduction across all datasets. It followed a near-optimal trajectory on MNIST and EMNIST and continued to reduce loss steadily on more complex tasks like Fashion-MNIST Figure 24 and CIFAR-10 Figure 25, where other strategies plateaued. By balancing selection across learning potential, battery cost, and privacy risk, RBPS maintained strong contributors in the loop longer, resulting in more sustained and efficient loss convergence.

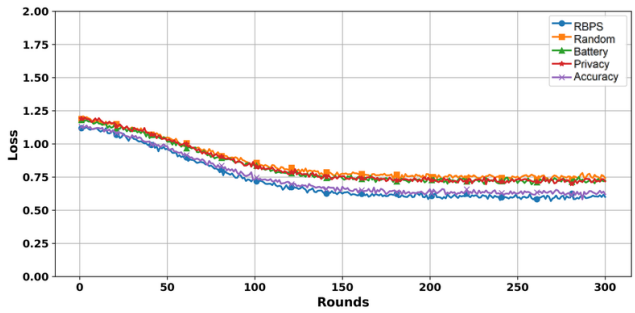

Figure 22. Loss over 300 rounds (MNIST).

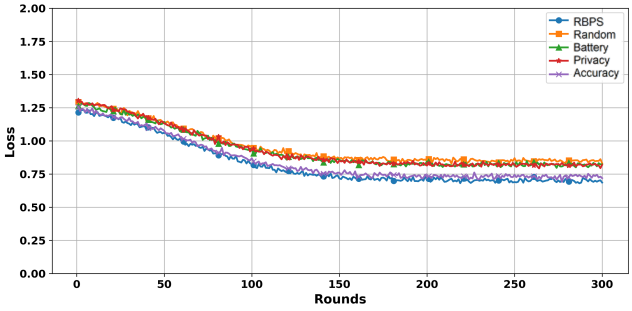

Figure 23. Loss over 300 rounds (EMNIST).

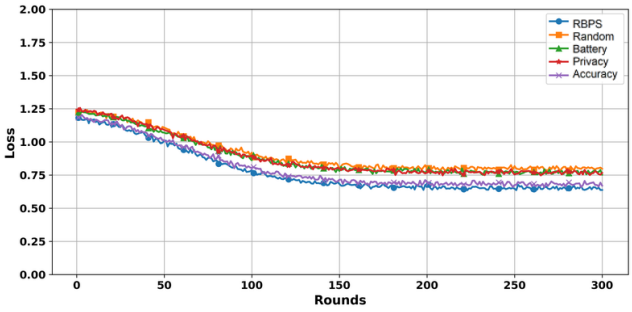

Figure 24. Loss over 300 rounds (Fashion-MNIST).

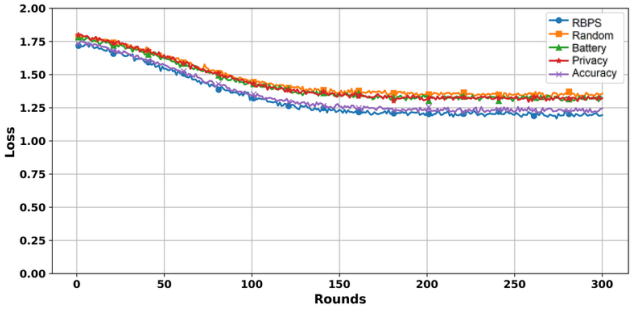

Figure 25. Loss over 300 rounds (CIFAR-10).

7.3.3. Battery Level over 300 Rounds

As shown in the battery consumption plots for MNIST Figure 26, EMNIST Figure 27, Fashion-MNIST Figure 28, and CIFAR-10 Figure 29, all methods begin with the same initial battery level and exhibit a significant drop during the early communication rounds. This behavior corresponds to the training phase where clients are actively participating in the FL process. After a certain point (which varies per dataset), the battery level decreases more slowly. This shift is due to the fact that although the server stops selecting certain clients for training, those clients continue performing peripheral tasks (e.g., monitoring, communication, or local computation), which still consume energy but at a reduced rate. Among the methods, "Battery" and "RBPS" consistently show better energy preservation, while "Accuracy" and "Random" lead to faster battery depletion. The overall trend confirms the trade-off between participation intensity and energy efficiency across different datasets.

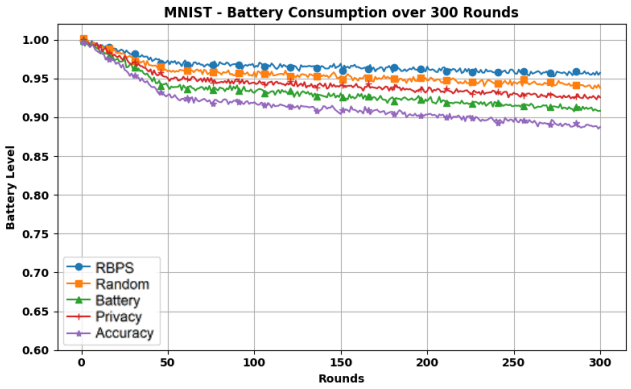

Figure 26. Battery level over 300 rounds (MNIST).

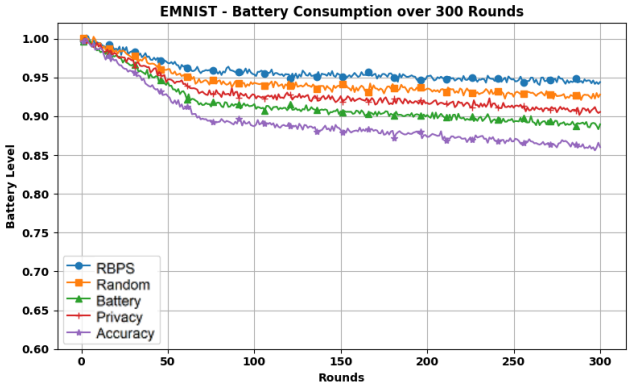

Figure 27. Battery level over 300 rounds (EMNIST).

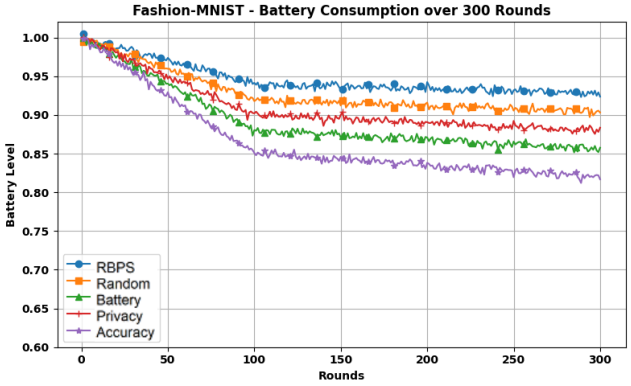

Figure 28. Battery level over 300 rounds (Fashion-MNIST).

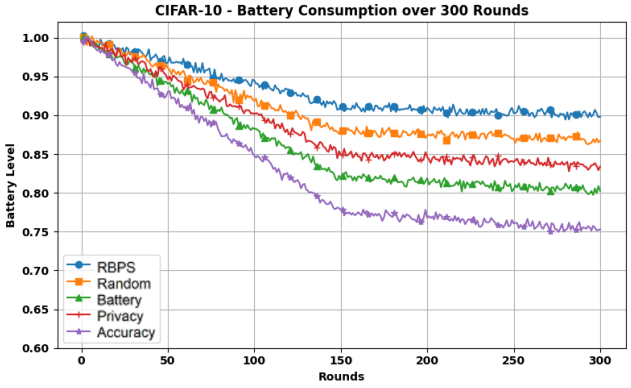

Figure 29. Battery level over 300 rounds (CIFAR-10).

7.3.4. Privacy Loss over 300 Rounds

The privacy loss Figures 30–33 (CIFAR-10) show that all methods start from the same initial privacy level and exhibit a noticeable decline during the initial training rounds. This decline is attributed to the fact that, during FL, participating clients share updated model parameters (weights and biases) with the server. Although raw data is not transmitted, these shared parameters can still reveal information about the local data distribution, leading to potential privacy leakage. After a dataset-specific round, the privacy loss curves stabilize. This stabilization occurs because the server stops selecting certain clients for training beyond that point. As a result, those clients no longer share updates, and their privacy loss remains constant. The method labeled "Privacy" shows the lowest overall privacy degradation, followed by "RBPS (Ours)", indicating stronger privacy-preserving behavior. In contrast, methods such as "Accuracy" and "Random" lead to higher cumulative privacy loss across all datasets.

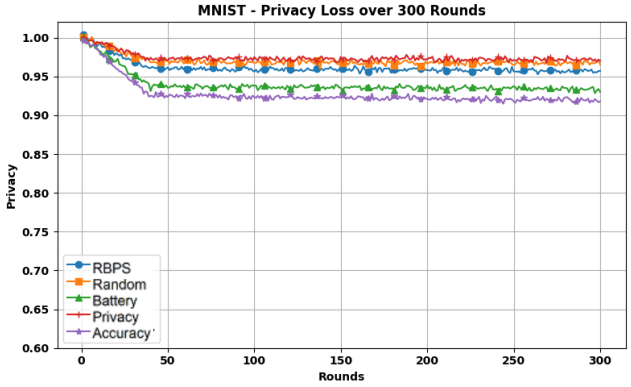

Figure 30. Privacy loss over 300 rounds (MNIST).

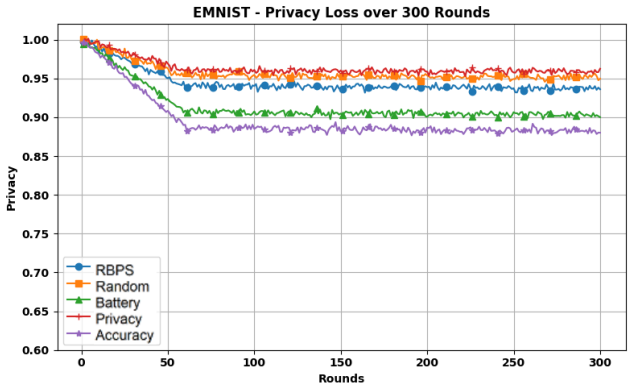

Figure 31. Privacy loss over 300 rounds (EMNIST).

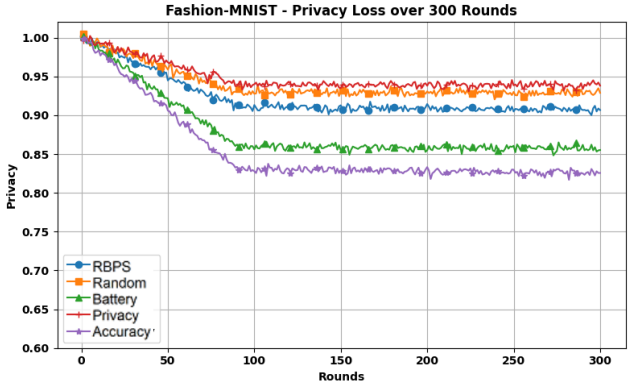

Figure 32. Privacy loss over 300 rounds (Fashion-MNIST).

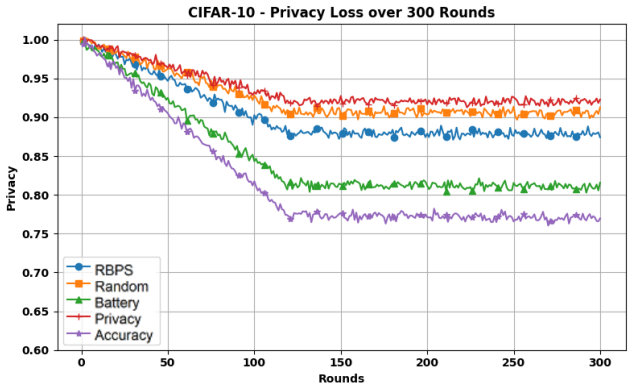

Figure 33. Privacy loss over 300 rounds (CIFAR-10).

7.4. Discussion

To ensure a fair and representative evaluation, we compare RBPS with four distinct client selection strategies: FedCS, Oort, FRS, and FedProm. These methods were chosen because they reflect fundamentally different priorities in FL system design, including resource awareness, accuracy optimization, fairness, and privacy personalization. Rather than comparing against every existing approach, we focused on methods with minimal overlap to highlight meaningful trade-offs and avoid redundancy.

Figures 34–36 illustrate the comparative results across 200 rounds of training. As shown, Oort achieves the fastest and highest accuracy convergence, but it does so at the cost of battery inefficiency and privacy leakage. FedProm demonstrates strong privacy preservation, outperforming all methods in that dimension, but with significantly lower accuracy. FRS and FedCS achieve moderate accuracy while maintaining efficient battery use, though they lack mechanisms for real-time privacy control. RBPS, in contrast, provides a robust balance: its accuracy closely approaches Oort, its privacy performance is near that of FedProm, and its battery consumption remains within sustainable bounds. These observations are summarized qualitatively in Table 7, which confirms that RBPS is the only method that performs consistently well across all three dimensions, making it a strong candidate for practical deployment in heterogeneous FL environments.

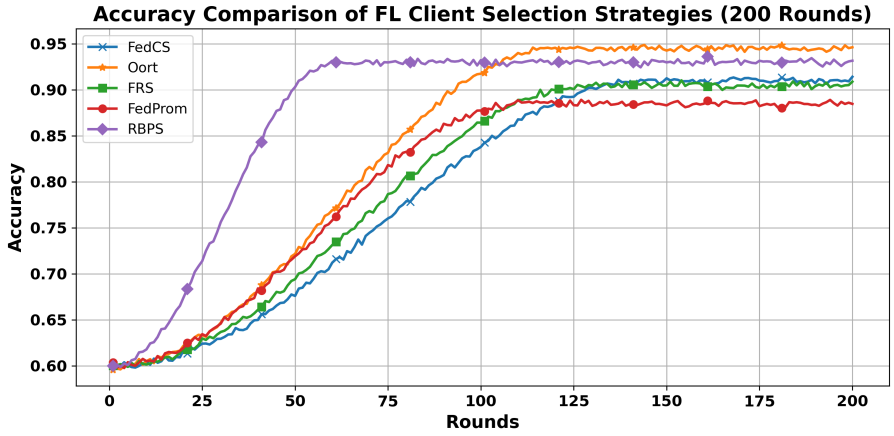

Figure 34. Model accuracy over 200 rounds on MNIST.

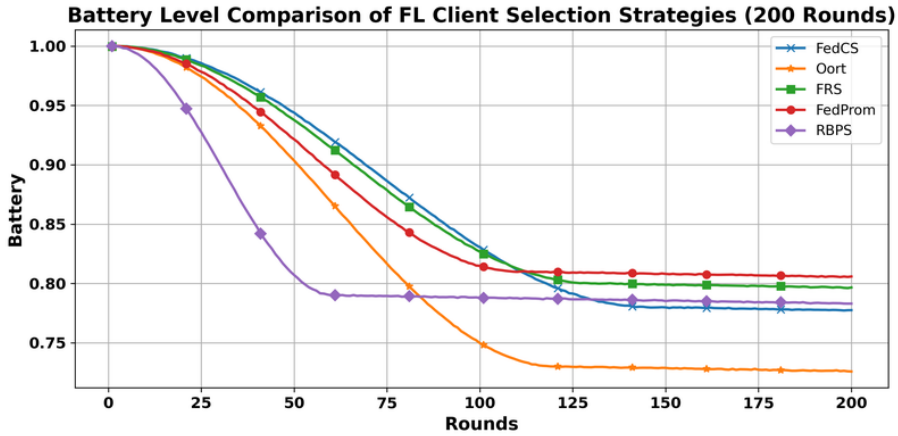

Figure 35. Battery level over 200 rounds on MNIST.

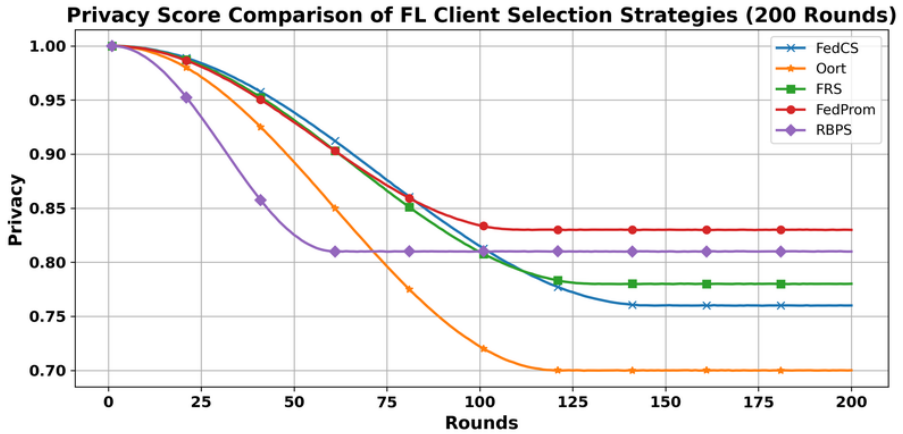

Figure 36. Privacy score over 200 rounds on MNIST.

Table 7. Comparison of FL client selection strategies vs. RBPS.

Method	Accuracy	Battery Efficiency	Privacy	Remarks
Oort	Very High	Poor	Poor	Fastest convergence; inefficient and privacy-leaking.
FedCS	Medium	Moderate	Moderate	Resource-matching; lacks privacy-awareness.
FRS	Medium	High	Moderate	Fair and battery-efficient; not privacy-aware.
FedProm	Low	High	Best	Maximizes privacy and personalization; lower accuracy.
RBPS (Ours)	High	Moderate	Strong	Balanced across all objectives; dynamic and cooperative.

8. Conclusions and Future Work

This paper presented RBPS, a centralized strategy with cooperative intent and an adaptive client selection strategy designed to address the challenges of heterogeneity, battery constraints, and privacy sensitivity in real-world learning environments. By integrating three critical factors (accuracy contribution, battery cost, and privacy sensitivity) into a dynamic reward-based scoring mechanism, RBPS enables the selection of suitable clients without overloading any specific device category. The strategy was evaluated using simulated pools of smartphones, edge devices, and IoT sensors across four benchmark datasets, under diverse client pool sizes and device distributions. Comparative experiments demonstrated that RBPS consistently achieves a more balanced trade-off between model performance, battery efficiency, and privacy preservation when compared to base-

line and SOA approaches. RBPS achieved an accuracy improvement of up to 25% over SOA methods such as Oort and FedCS on the MNIST dataset. When compared to single-metric baselines across all datasets, RBPS provided better balance among accuracy, energy consumption, and privacy impact. In heterogeneous client pools, the method preserved model stability while reducing battery usage variance and limiting privacy exposure. As the number of clients increased, RBPS maintained convergence with minimal degradation in accuracy, demonstrating robustness and scalability in complex FL scenarios. While this study is based on detailed simulation modeling, all operational parameters including battery consumption patterns, device heterogeneity, and privacy risk factors were derived from realistic hardware characteristics and empirically informed system behavior. While the current study relies on simulation-based evaluation, which is standard in early-stage FL research, we acknowledge that further validation on physical hardware is an important next step. Implementing RBPS on real embedded and mobile devices will allow us to observe potential runtime effects related to hardware-level variability, resource usage, and network dynamics. This transition from simulation to deployment is part of our roadmap and will strengthen the practical applicability of our approach. In future work, we aim to deploy RBPS on a heterogeneous testbed composed of embedded and mobile devices to empirically measure key performance indicators such as per-round battery consumption, communication latency, and memory utilization. Furthermore, we will evaluate privacy exposure by analyzing model updates for vulnerability to data inference through controlled adversarial probing, thereby assessing the actual leakage risk associated with different device classes. We also plan to develop a formal convergence analysis of RBPS by framing the client-selection game as a potential game and proving the existence (and uniqueness) of a NE under our reward structure and to quantify long-term fairness and client sustainability by computing metrics such as Jain's fairness index across training rounds and, if needed, augment RBPS with fairness regularization to mitigate any systematic under sampling of specific device groups. These experiments will provide a comprehensive validation of RBPS under live network conditions and offer insights into its effectiveness and robustness in real-world FL deployments.

Author Contributions: Conceptualization, Z.D. and M.M.; methodology, M.M.; software, Z.D.; validation, M.M.; formal analysis, Z.D.; investigation, Z.D.; resources, M.M.; data curation, Z.D.; writing—original draft preparation, Z.D.; writing, review and editing, M.M.; visualization, Z.D.; supervision, M.M.; project administration, M.M.; funding acquisition, M.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable. This study did not involve human participants or animals.

Informed Consent Statement: Not applicable. This study did not involve human participants.

Data Availability Statement: The datasets used in this study are publicly available. MNIST is available at http://yann.lecun.com/exdb/mnist/, (accessed on 3 May 2025). EMNIST is available at https://www.nist.gov/itl/products-and-services/emnist-dataset, (accessed on 3 May 2025). Fashion-MNIST is available at https://github.com/zalandoresearch/fashion-mnist, (accessed on 3 May 2025). The CIFAR-10 dataset is available at https://www.cs.toronto.edu/~kriz/cifar.html, (accessed on 3 May 2025).

Conflicts of Interest: Author Massimo Merenda's affiliation with the company HWA srl Spin-off Unirc reflects the ownership of a share in the company. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as potential conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

FL	Federated learning
RBPS	Reward-based payoff strategy
ML	Machine learning
NE	Nash equilibrium
SOA	State-of-the-art
FedCS	Federated client selection
FRS	Fair resource scheduling
MDA	Multi device aggregation
TiFL	Tiered FL
FedAvg	Federated Averaging
SGD	Stochastic gradient descent
GT	Game theory
NN	Neural network
Flwr	Flower
CNN	Convolutional neural network

References

1. Yang, Q.; Liu, Y.; Chen, T.; Tong, Y. Federated Machine Learning: Concept and Applications. *ACM Trans. Intell. Syst. Technol.* **2019**, *10*, 1–19. [CrossRef]
2. Cho, J.; Kim, H. Towards Efficient Client Selection in Federated Learning: A Survey. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *33*, 1621–1634.
3. Tan, Y.; Long, G.; Liu, L.; Zhou, T.; Lu, Q.; Jiang, J.; Zhang, C. FedProto: Federated Prototype Learning Across Heterogeneous Clients. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtual, 22 February–1 March 2022; Volume 36, pp. 8432–8440.
4. Ndou, N.; Ajoodha, R.; Jadhav, A. Music Genre Classification: A Review of Deep-Learning and Traditional Machine-Learning Approaches. In Proceedings of the 2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS), Toronto, ON, Canada, 21–24 April 2021; pp. 1–6.
5. Wang, P.; Fan, E.; Wang, P. Comparative Analysis of Image Classification Algorithms Based on Traditional Machine Learning and Deep Learning. *Pattern Recognit. Lett.* **2021**, *141*, 61–67. [CrossRef]
6. Nishio, T.; Yonetani, R. Client selection for federated learning with heterogeneous resources in mobile edge. In Proceedings of the ICC 2019—2019 IEEE International Conference on Communications (ICC), Shanghai, China, 20–24 May 2019; pp. 1–7.
7. Lai, K.; Bennis, M.; Niyato, D. Oort: Efficient Federated Learning via Guided Participant Selection. *IEEE Trans. Mob. Comput.* **2021**, *20*, 3305–3318.
8. Sultana, A.; Haque, M.M.; Chen, L.; Xu, F.; Yuan, X. Eiffel: Efficient and Fair Scheduling in Adaptive Federated Learning. *IEEE Trans. Parallel Distrib. Syst.* **2022**, *33*, 4282–4294. [CrossRef]
9. Niu, X.; Sun, L.; Song, L.; Li, B. FedProm: Preference- and priority-aware client selection for personalized federated learning. In Proceedings of the ACM International Conference on Multimedia (MM), Lisboa, Portugal, 10–14 October 2022; pp. 4345–4353.
10. Liu, Y.; Yang, Y.; Zhang, Y.; Wang, W. FedGCS: Federated graph client selection in edge computing. In Proceedings of the IEEE International Conference on Communications (ICC), Dublin, Ireland, 7–11 June 2020.
11. Diao, Q.; Meng, T.; Xiao, W.; Li, M.; Chen, J.; Zhang, M. HeteroFL: Computation and communication efficient federated learning for heterogeneous clients. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 4290–4300.
12. Chai, Z.; Ali, A.; Zawad, S.; Truex, S.; Anwar, A.; Baracaldo, N.; Zhou, Y.; Ludwig, H.; Yan, F.; Cheng, Y. Tifl: A Tier-Based Federated Learning System. In Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing (HPDC), Stockholm, Sweden, 23–26 June 2020; pp. 125–136.
13. Li, Q.; Li, X.; Zhou, L.; Yan, X. Adafl: Adaptive client selection and dynamic contribution evaluation for efficient federated learning. In Proceedings of the ICASSP 2024—2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Seoul, Republic of Korea, 14–19 April 2024; pp. 6645–6649.
14. Zaw, C.W.; Pandey, S.R.; Kim, K.; Hong, C.S. Energy-aware resource management for federated learning in multi-access edge computing systems. *IEEE Access* **2021**, *9*, 34938–34950. [CrossRef]
15. Xu, Y.; Xiao, M.; Wu, J.; Tan, H.; Gao, G. A personalized privacy preserving mechanism for crowdsourced federated learning. *IEEE Trans. Mob. Comput.* **2023**, *23*, 1568–1585. [CrossRef]

16. Dritsas, E.; Trigka, M. Federated Learning for IoT: A Survey of Techniques, Challenges, and Applications. *J. Sens. Actuator Netw.* **2025**, *14*, 9. [CrossRef]
17. Liu, Y.; Qu, Z.; Wang, J. Compressed Hierarchical Federated Learning for Edge-Level Imbalanced Wireless Networks. *IEEE Trans. Comput. Soc. Syst.* **2025**, 1–12. [CrossRef]
18. McMahan, H.B.; Moore, E.; Ramage, D.; Hampson, S.; Agüera y Arcas, B. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS), Fort Lauderdale, FL, USA, 20–22 April 2017; pp. 1273–1282.
19. Li, T.; Sahu, A.K.; Talwalkar, A.; Smith, V. Federated Optimization in Heterogeneous Networks. In Proceedings of the Machine Learning and Systems (MLSys), Austin, TX, USA, 2–4 March 2020; Volume 2, pp. 429–450.
20. Marfo, W.; Tosh, D.K.; Moore, S.V. Adaptive client selection in federated learning: A network anomaly detection use case. *arXiv* **2025**, arXiv:2501.15038.
21. Liu, J.; Sun, H.; Xu, H. Bayesian Nash Equilibrium in price competition under multinomial logit demand. *Eur. J. Oper. Res.* **2025**, *324*, 669–689. [CrossRef]
22. Facchinei, F.; Kanzow, C. Generalized Nash equilibrium problems. *Ann. Oper. Res.* **2010**, *175*, 177–211. [CrossRef]
23. Armoogum, S.; Bassoo, V. Privacy of energy consumption data of a household in a smart grid. In *Smart Power Distribution Systems*; Elsevier: Amsterdam, The Netherlands, 2019; pp. 163–177.
24. Klass, A.B.; Wilson, E.J. Remaking energy: The critical role of energy consumption data. *Calif. Law Rev.* **2016**, *104*, 1095.
25. Espressif Systems. ESP32 Series Datasheet [Online]. 2022. Available online: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf (accessed on 2 July 2024).
26. Li, T.; Sahu, A.K.; Talwalkar, A.; Smith, V. Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Process. Mag.* **2020**, *37*, 50–60. [CrossRef]
27. Rahmati, A.; Zhong, L. Understanding human–battery interaction on mobile phones. In Proceedings of the 9th International Conference on Human Computer Interaction with Mobile Devices and Services, Singapore, 9–12 September 2007; pp. 265–272.
28. Fredrikson, M.; Jha, S.; Ristenpart, T. Model inversion attacks that exploit confidence information and basic countermeasures. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, 12–16 October 2015; pp. 1322–1333.
29. Shokri, R.; Stronati, M.; Song, L.; Shmatikov, V. Membership inference attacks against machine learning models. In Proceedings of the IEEE Symposium on Security and Privacy, San Jose, CA, USA, 22–26 May 2017; pp. 3–18.
30. Katsomallos, M.; Tzompanaki, K.; Kotzinos, D. Privacy, space and time: A survey on privacy-preserving continuous data publishing. *J. Spat. Inf. Sci.* **2019**, *19*, 57–103. [CrossRef]
31. Zhou, J.; Cao, Z.; Dong, X.; Vasilakos, A.V. Security and privacy for cloud-based IoT: Challenges. *IEEE Commun. Mag.* **2017**, *55*, 26–33. [CrossRef]
32. NVIDIA Corporation. Jetson Nano Developer Kit User Guide [Online]. 2021. Available online: https://developer.nvidia.com/embedded/jetson-nano-developer-kit (accessed on 3 August 2024).
33. Google LLC. Pixel 6 Battery Specifications [Online]. 2022. Available online: https://store.google.com/product/pixel_6_specs (accessed on 10 August 2024).
34. Adafruit Industries. Adafruit: Electronics, DIY Electronics, and Open-Source Hardware. Available online: https://www.adafruit.com/ (accessed on 3 May 2025).
35. NVIDIA Corporation, Jetson Modules, Support, Ecosystem, and Lineup. Available online: https://developer.nvidia.com/embedded/jetson-modules (accessed on 3 May 2025).
36. Samsung Electronics Italia S.p.A. Samsung Italia: Smartphone, Elettrodomestici, TV, Informatica. Available online: https://www.samsung.com/it/ (accessed on 3 May 2025).
37. Google LLC. Google Store Italia: Dispositivi e Accessori Made by Google. Available online: https://store.google.com/?hl=it (accessed on 3 May 2025).