Titre: Title:	Game-Theoretic Client Selection Mechanisms for Fairness, Incentives, and Security in Federated Learning with Non-IID Data
Auteur: Author:	Sarhad Arisdakessian
Date:	2025
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Arisdakessian, S. (2025). Game-Theoretic Client Selection Mechanisms for Fairness, Incentives, and Security in Federated Learning with Non-IID Data [Thèse de doctorat, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/71046/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/71046/
Directeurs de recherche: Advisors:	Omar Abdul Wahab
Programme: Program:	génie informatique

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Game-Theoretic Client Selection Mechanisms for Fairness, Incentives, and Security in Federated Learning with Non-IID Data

SARHAD ARISDAKESSIAN
Département de génie informatique et génie logiciel

Thèse présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*
Génie informatique

Décembre 2025

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Cette thèse intitulée :

Game-Theoretic Client Selection Mechanisms for Fairness, Incentives, and Security in Federated Learning with Non-IID Data

présentée par **Sarhad ARISDAKESSIAN**
en vue de l'obtention du diplôme de *Philosophiæ Doctor*
a été dûment acceptée par le jury d'examen constitué de :

DEDICATION

Throughout my academic journey, from my earliest years of study to the completion of this thesis, my family has been my constant source of strength, patience, and unconditional support. This work is, above all, dedicated to them.

To my parents, whose sacrifices laid the foundation for everything I have achieved. Their belief in the value of education, their resilience through hardship, and their unwavering encouragement shaped not only my academic path but also the person I have become. Every step forward in this journey carries the imprint of their efforts, values, and love.

A very special thank you goes to my wife. Her patience, understanding, and constant support carried me through the most demanding moments of this journey. She stood beside me through long hours, uncertainty, and pressure, offering encouragement, calm, and strength when it was needed most. This thesis would not have been possible without her presence, belief, and unwavering support.

To my family, who stood by me during the long years of study, uncertainty, and persistence that research demands. Your understanding, encouragement, and quiet confidence sustained me through moments of doubt and fatigue, reminding me why this journey mattered. You shared in the challenges, celebrated the milestones, and never stopped believing in me.

This thesis is a reflection not only of individual effort, but of a collective support system built on love, trust, and sacrifice. Whatever merit it holds belongs as much to my family as it does to me.

To you, I owe more than words can express.

ACKNOWLEDGEMENTS

I would like to extend my deepest gratitude to my advisor, Dr. Omar Abdul Wahab, for his unwavering support, insightful guidance, and countless revisions throughout my doctoral journey. This achievement would not have been possible without his dedication and encouragement.

I am also sincerely thankful to Dr. Azzam Mourad and Dr. Hadi Otrok for their continued support and collaborative spirit. It was a privilege to work alongside them.

Finally, a special thank you to Osama Wehbi for his honest, kind, and invaluable assistance throughout our work.

RÉSUMÉ

L'apprentissage fédéré (FL) s'est imposé comme un paradigme prometteur permettant l'entraînement collaboratif de modèles de machine learning à travers des silos de données décentralisés, offrant aux organisations la possibilité de bénéficier d'une intelligence collective tout en préservant la confidentialité des données. Malgré ce potentiel, l'efficacité et la durabilité du FL dans des déploiements réels restent limitées par plusieurs défis critiques. Ceux-ci incluent la participation peu fiable ou malhonnête des clients, les déséquilibres en matière de ressources disponibles, l'absence de mécanismes d'incitation transparents, ainsi que des hypothèses irréalistes concernant la distribution des données. Cette thèse s'attaque à ces limitations au moyen de quatre cadres indépendants mais complémentaires, chacun abordant une dimension particulière de l'écosystème FL : la gestion de la confiance, l'inclusivité fondée sur les coalitions, les marchés incitatifs compatibles, et le benchmarking réaliste. Plusieurs de ces cadres s'appuient sur des modèles de théorie des jeux, notamment les jeux de coalition, de Stackelberg et les enchères inversées.

Le premier cadre introduit un mécanisme de sélection des clients fondé sur la confiance, modélisé à travers le prisme des jeux de coalition et des jeux hédoniques. Dans ce contexte, les clients forment implicitement des coalitions avec l'agrégateur, où la valeur de l'adhésion dépend à la fois de la fiabilité individuelle (confiance objective) et de la perception de la fiabilité des pairs (confiance subjective). En combinant un suivi basé sur les ressources avec des préférences guidées par la réputation, ce cadre capture la nature hédonique de la formation des coalitions : les clients préfèrent participer lorsqu'ils sont regroupés avec des pairs fiables, tandis que les participants malhonnêtes ou de faible qualité sont exclus. Cette interprétation par la théorie des jeux met en évidence la manière dont des coalitions stables peuvent être soutenues par les dynamiques de confiance, réduisant ainsi l'impact des passagers clandestins. Les expériences montrent que l'intégration d'un raisonnement coalitionnel fondé sur la confiance améliore non seulement la convergence, mais renforce également l'équité et la robustesse dans des conditions hétérogènes et adversariales.

Le deuxième cadre, StackFed, traite les inégalités structurelles dans la participation des clients. Dans de nombreux systèmes FL, les clients limités en ressources sont marginalisés, ce qui restreint à la fois l'inclusivité et la diversité des données contribuant à l'entraînement. Pour contrer ce phénomène, StackFed emploie une approche de jeu de Stackelberg, dans laquelle des clients riches en ressources jouent le rôle de leaders et forment des coalitions avec des clients plus faibles, dits suiveurs. Les leaders définissent les conditions de coalition, tandis que les suiveurs choisissent stratégiquement leurs alignements, créant ainsi des coalitions stables de type leader–suiveur qui démocratisent l'accès à l'entraînement. Cette conception favorise l'équité en évitant l'exclusion des clients faibles et améliore l'efficacité de l'apprentissage en équilibrant les charges computationnelles. Les résultats expérimentaux montrent que StackFed accroît non seulement l'inclusivité, mais améliore aussi la capacité de généralisation en intégrant les données de clients qui seraient autrement sous-utilisés. Le troisième cadre présente une architecture de marché pour l'apprentissage fédéré, qui formalise les interactions économiques entre les Propriétaires de Tâches, les Propriétaires de Réseau et les Clients. Dans ce modèle, les Propriétaires de Tâches définissent les exigences des modèles ainsi que les budgets, les Propriétaires de Réseau orchestrent l'entraînement, et les Clients entrent en compétition pour contribuer aux mises à jour. Le marché emploie des mécanismes d'enchères inversées pour allouer les tâches et intègre un système de réputation dynamique afin d'assurer une responsabilité à long terme. Les Clients sont récompensés proportionnellement à la qualité de leur contribution, les Propriétaires de Tâches bénéficient de mises à jour fiables et rentables, et les Propriétaires de Réseau soutiennent leurs opérations grâce à des commissions équitables. Ce cadre montre comment les dimensions économiques et techniques peuvent être unifiées pour garantir la durabilité. Les expériences valident que ce marché atteint une meilleure efficacité et une plus grande équité comparé à des systèmes d'incitation ad hoc, tout en décourageant la participation malhonnête ou à faible effort.

Enfin, le quatrième cadre propose une stratégie de partitionnement hiérarchique basée sur Dirichlet à deux niveaux, sensible aux caractéristiques, pour le benchmarking des distributions non-IID. Les méthodes de partitionnement traditionnelles échouent souvent à capturer l'hétérogénéité complexe des ensembles de données réels, limitant ainsi la validité externe des expériences FL. La méthode proposée introduit deux niveaux de contrôle distributionnel : (i) un Dirichlet au niveau des clusters, qui partitionne les données en fonction de regroupements de caractéristiques latentes, et (ii) un Dirichlet au niveau des classes, qui régule l'allocation des étiquettes au sein des clusters. Cette approche hiérarchique génère des ensembles de données clients à la fois synthétiques et réalistes, reflétant à la fois la rareté inter-clients et l'asymétrie intra-client. En s'appuyant sur ce cadre de benchmarking, les chercheurs peuvent tester plus précisément les algorithmes FL dans des conditions reflétant l'hétérogénéité réelle. Les expériences sur divers ensembles de données montrent que le Dirichlet à deux niveaux reproduit mieux le caractère non-IID du monde réel que le partitionnement Dirichlet classique à un seul niveau.

Pris ensemble, ces cadres font progresser l'apprentissage fédéré selon quatre dimensions : la confiance, l'inclusivité, l'alignement incitatif et le réalisme. Bien que chaque contribution ait une valeur indépendante, elles forment collectivement une feuille de route pour surmonter les principales limitations du FL. La thèse démontre qu'en s'attaquant à ces défis systémiques, le FL peut évoluer d'un concept prometteur à un paradigme pratique et durable.

ABSTRACT

Federated learning (FL) has emerged as a promising paradigm for enabling collaborative machine learning across decentralized data silos, allowing organizations to benefit from collective intelligence while preserving data privacy. Despite this potential, the effectiveness and sustainability of FL deployments remain constrained by critical challenges. These include unreliable or dishonest client participation, imbalances in resource availability, and the absence of transparent incentive mechanisms. This thesis tackles these limitations through four independent but complementary frameworks, each addressing a distinct dimension of the FL ecosystem: trust management, coalition-based inclusivity, incentive-compatible marketplaces, and realistic benchmarking. Several of these frameworks are grounded in game-theoretic models, including coalitional, Stackelberg, and reverse auction games.

The first framework introduces a trust-aware client selection mechanism modeled through the lens of coalitional and hedonic games. In this setting, clients implicitly form coalitions with the aggregator, where the value of joining a coalition depends both on individual trustworthiness (objective trust) and the perceived reliability of peers (subjective trust). By blending resource-based monitoring with reputation-driven preferences, the framework captures the hedonic nature of coalition formation: clients prefer to participate when they are grouped with trusted peers, while low-quality participants are excluded. This game-theoretic interpretation highlights how stable coalitions can be sustained through trust dynamics, mitigating the impact of selfish clients. Experiments demonstrate that incorporating such trust-aware coalitional reasoning, not only improves convergence but also enhances fairness during the FL process.

The second framework addresses the structural inequalities in client participation. In many FL systems, resource-constrained clients are marginalized, limiting both the inclusivity and the diversity of contributed data. To counter this, the approach employs a Stackelberg game-theoretic approach, in which resource-rich leader clients form coalitions with less-resourceful follower clients. Leaders set coalition terms, while followers strategically choose alignments, creating stable leader–follower coalitions that democratize access to training epochs or training process. This design improves fairness by ensuring less-resourceful clients are not excluded, while also improving learning efficiency. Experimental results show that the proposed approach not only enhances inclusivity but also leads to stronger generalization performance by incorporating data from clients who would otherwise remain underutilized.

The third framework presents a federated learning marketplace architecture that formalizes economic interactions among different stakeholders. In this setting, Task Owners specify model requirements and budgets, Network Owners orchestrate training, and Clients compete to contribute updates. The marketplace employs reverse auction mechanisms to allocate tasks and integrates a dynamic reputation system to ensure long-term accountability. Clients are rewarded for their contribution, Task Owners receive cost-effective and reliable updates, and Network Owners sustain operations through fair commissions. This framework demonstrates how economic and technical dimensions can be unified to ensure sustainability. Experiments validate that the marketplace achieves higher efficiency and fairness compared to ad hoc incentive schemes, while discouraging dishonest or low-effort participation.

Finally, the fourth framework contributes a feature-aware two-level Dirichlet partitioning strategy for benchmarking non-IID data distributions. Traditional partitioning methods often fail to capture the complex heterogeneity of real-world datasets, limiting the external validity of FL experiments. The proposed method introduces two levels of distributional control: (i) a cluster-level Dirichlet that partitions data by latent feature clusters, and (ii) a class-level Dirichlet that governs the allocation of labels within clusters. This hierarchical approach generates synthetic yet realistic client datasets that reflect both inter-client sparsity and intra-client skewness. By benchmarking against this framework, researchers can more accurately stress-test FL algorithms under conditions that mirror real-world heterogeneity. Experiments across datasets highlight that the two-level Dirichlet produces distributions that better capture non-IIDness compared to single-level Dirichlet partitioning.

Together, these frameworks advance federated learning along the dimensions of trust, inclusivity, incentive alignment, democratization, and realism. While each contribution stands independently, together they form a holistic agenda for overcoming limitations in FL. The thesis demonstrates that by addressing such systemic challenges, FL can evolve into a practical and sustainable paradigm.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

In this chapter, we present the background of our research, outline the challenges addressed in this thesis, formulate the research questions, and define the objectives of our study, alongside discussing our contributions.

1.1 Research Context and Problem Statement

The rapid rise in IoT and smart devices adoption generates vast amounts of data due to their sensing, storing, and transmitting capabilities. Traditionally, this data needed to be transferred to a central repository for analysis, sparking significant privacy concerns, as users are uncomfortable with sensitive data being sent to third-party servers, especially after major hacking incidents exposing private information. Federated Learning (FL) addresses these concerns by allowing local training on devices; only the model updates, not the raw data, are sent to a federated server [1] (FS). The server then aggregates these updates to create an improved global model, enabling clients to benefit from collective insights without sharing their raw private data.

Since the inception of FL [1] [2], it has rapidly evolved to tackle critical challenges in the realm of distributed learning, such as data heterogeneity [3], communication constraints [4], and privacy preservation [5]. However, despite these advances, FL still faces obstacles [6] [7] [8] especially in client selection mechanisms, where efficiency of the training is largely dependent on the system and data resources of individual devices. We summarize some of these challenges as follows:

1. **Client Heterogeneity:** FL relies on end devices with widely varying system capabilities, including differences in CPU power, memory, battery life, and communication bandwidth. Resource-rich devices can complete local training efficiently, whereas resource-constrained devices may lag behind, slowing down aggregation and creating stragglers that take significantly longer than others to complete local training or communication, thereby delaying the aggregation and slowing down overall round completion. Such disparities not only reduce the efficiency of global training but also risk excluding weaker devices, leading to systematic bias in which only high-capacity clients influence the model.

[1]Used interchangeably with the term ‘central server’

2. **Non-IID and Imbalanced Data:** Unlike centralized learning settings, FL operates on local datasets that reflect the natural data generation patterns of users or systems. As a result, data across clients is often *non-IID* (non-independent and identically distributed): skewed class distributions, imbalanced sample quantities, and varying data quality are the norm. This heterogeneity creates biased model updates that hinder global convergence, degrade accuracy, and reduce the representativeness of the trained model. Addressing this is crucial to ensure that FL produces models that generalize across diverse real-world scenarios.

3. **Selfish and Free-Riding Clients:** In practice, some clients may behave selfishly by contributing minimal effort or even dishonest updates in order to conserve their own computational and communication resources. Such free-riding behavior reduces the overall quality of the aggregated model while still receiving the latest model updates. Without mechanisms to detect and penalize dishonest or low-effort participation, FL systems remain vulnerable to reduced efficiency.

4. **Trust and Client Autonomy:** Current FL protocols typically grant the central server extensive authority in selecting clients for training rounds. While this ensures some level of coordination, it undermines the autonomy of clients and creates a centralized power imbalance in what is otherwise intended to be a decentralized system. Furthermore, trust relationships in FL are rarely mutual: clients are expected to trust the server, but mechanisms for clients to evaluate the trustworthiness of the system or their peers are underdeveloped. This discourages participation and raises concerns about transparency and fairness.

5. **Incentive Limitations and Participation Sustainability:** Long-term sustainability of FL requires active and recurring participation from diverse clients. However, without carefully designed incentive mechanisms, many clients may lack motivation to participate, particularly when participation incurs resource costs with little immediate benefit. Existing incentive schemes often fail to account for long-term engagement, fairness among heterogeneous contributors, or protection against dishonest behavior, mainly because of their focus on immediate performance-based rewards, neglecting sustained participation dynamics, contribution diversity, and trust evaluation over time. This lack of robust, incentive-compatible design threatens the viability of FL in real-world deployments where client motivation cannot be assumed.

1.2 Motivation

The challenges outlined in section 1.1 motivate the need for approaches that move beyond conventional client selection in the realm of FL. While prior research has proposed partial solutions, several fundamental gaps remain that hinder the real-world applicability of federated learning.

First, trust, reputation, and accountability are essential for any collaborative learning system, yet current frameworks assume honest participation. Without mechanisms to evaluate and enforce trust, FL systems remain vulnerable to free-riders and dishonest contributors, which reduce both the efficiency and the perceived fairness of participation. A robust solution must explicitly integrate trust modeling and reputation into the client selection process.

Second, client inclusivity and fairness remain integral in client selection. Resource heterogeneity often leads to the exclusion of weaker clients, reinforcing systemic inequalities in participation and limiting the diversity of data that informs the global model. Ensuring fair opportunities for participation, regardless of device capabilities, is critical to building models that generalize across diverse populations.

Third, there is the need for incentive-compatible mechanisms that promote sustained engagement. Existing incentive schemes are either simplistic, fail to account for long-term participation dynamics, and do not formally separate the different stakeholders in FL. In practical deployments, clients must be motivated not only to join but also to contribute meaningfully across training tasks. Incentive mechanisms that balance fairness, contribution quality, and sustainability are vital to the long-term viability of FL.

Fourth, there is a need of realistic benchmarking tools for evaluating client selection and incentive mechanisms. Much of the existing research relies on simplified data partitioning strategies that fail to capture the full complexity of non-IID data distributions in practice. More realistic simulation frameworks are necessary to stress-test client selection policies and validate their robustness under heterogeneous and real-world conditions.

Taken together, these factors motivate a comprehensive research agenda that explores federated learning from multiple complementary perspectives: trust modeling, coalition formation, game-theoretic incentive design, and data-driven benchmarking. Addressing these challenges is important for advancing the foundations of FL, enabling its adoption in different domains and ecosystems.

1.3 Research Questions and Objectives

The challenges identified in Section 1.1 and the motivations outlined in Section 1.2 highlight the need for a systematic investigation into how federated learning can be fairer, more inclusive, and sustainable. To structure this investigation, this thesis formulates a set of research questions (RQs) that directly address the identified gaps. Each research question is accompanied by a corresponding research objective (Obj), which defines the actionable goal pursued in this work. Together, the questions and objectives provide a coherent roadmap for the frameworks developed and evaluated in this thesis.

1. **RQ1:** How can federated learning systems incorporate trust to discourage selfish, free-riding and low-quality participation and simultaneously democratize client involvement by enabling participants to join coalitions of their choice?

   Obj1: To develop a framework that integrates trust and client autonomy into federated learning, ensuring reliable participation while promoting fairness, inclusivity, and coalition-based collaboration.

2. **RQ2:** How can heterogeneous clients with uneven resources collaborate in a way that promotes inclusivity and fairness in federated learning?

   Obj2: To design a framework that enables cooperative structures among clients of varying resource capabilities, ensuring balanced participation and equitable contribution to global model training.

3. **RQ3:** How can federated learning ecosystems be structured to provide transparent incentives and sustain long-term participation across all stakeholders?

   Obj3: To establish an incentive-driven model that aligns the interests of all parties in federated learning, ensuring good reputation for the participants, fairness, accountability, and sustainable collaboration over time.

4. **RQ4:** How can we generate realistic benchmarks to evaluate the robustness of client selection and incentive mechanisms under heterogeneous non-IID data conditions?

   Obj4: To create a data simulation distribution framework that produces parameterized non-IID scenarios, enabling systematic evaluation of federated learning methods under diverse data conditions.

1.4 Contributions

Based on the above, this thesis makes four major contributions to the field of federated learning, each addressing a different challenge. Each of these four independent yet complementary frameworks is grounded in distinct theoretical models and validated through experiments.

Accordingly, the contributions of this thesis are summarized as follows:

1. **A client selection mechanism with trust and coalition autonomy (Chapter 3):** In our first contribution, we propose a framework that strengthens federated learning through the integration of trust and autonomy in client participation [9]. Clients are evaluated based on both objective measures and subjective trust, allowing the system to discourage dishonest or selfish participants. At the same time, the framework democratizes federated learning by enabling clients to form or join coalitions based on their preferences and perceived utilities. This dual emphasis on reliability and autonomy ensures that client selection is both robust and inclusive, improving fairness, and resilience against free-riding behaviors. This work targets and answers research question and objective 1.

2. **A framework that forms leader–follower coalitions to enhance fairness and give less resourceful clients participation opportunity (Chapter 4):** The second contribution presents a framework that organizes heterogeneous clients into partitions, ensuring equitable participation despite disparities in computational or data resources using Stackelberg games [10]. By structuring clients into leader–follower coalitions, where resource-rich leaders coordinate with resource-constrained followers, the framework democratizes access to training opportunities and reduces systemic exclusion. This approach improves fairness, strengthens model generalization, and enables stable participation in FL. This work targets and answers research question and objective 2.

3. **A Framework that establishes a federated marketplace with incentive and reputation mechanisms to ensure fair rewards, accountability, and long-term participation (Chapter 5):** In our third contribution, we present a federated learning marketplace model that structures economic interactions among different stakeholders in an FL environment, namely: Task Owners (entities who have learning tasks to accomplish), Network Owners (Orchestrators of FL, aggregators), and Clients. The marketplace applies reverse auction based mechanisms to allocate tasks under budget constraints and introduces a reputation system to reward long-term reliability. Through

the design of incentive-compatible mechanisms, this contribution fosters sustainable collaboration, fair compensation, and transparency among stakeholders. This work targets and answers research question and objective 3.

4. **A technique that generates realistic non-IID data distributions to enable fair benchmarking of federated learning strategies (Chapter 6):** In our fourth contribution, we propose a technique that improves data distribution modeling and enables more reliable FL evaluation under heterogeneous conditions. A two-level distribution framework is proposed that models both feature-level and class-level heterogeneity [11], generating modular and parameterized client datasets. This approach captures inter-client sparsity and intra-client skewness more accurately than traditional partitioners. By providing a flexible and statistically principled tool, this contribution enables rigorous evaluation of FL strategies under diverse non-IID data settings. This work targets and answers research question and objective 4.

Taken together, these contributions advance FL along various dimensions, such as: reliability, inclusivity, sustainability, and realism. Even though each framework stands independently as a solution to a challenge, collectively they form a comprehensive research agenda for building FL ecosystems that are technically sound, economically viable, and socially equitable.

Several chapters of this thesis are based on previously submitted and published articles. For coherence and clarity, the material has been revised and harmonized to fit within a unified dissertation framework. These revisions do not alter the original scientific contributions but aim to provide a clearer and more integrated presentation.

CHAPTER 2 BACKGROUND & LITERATURE REVIEW

In this chapter, we introduce the foundational concepts that form the basis of the thesis and present a systematic review of the literature related to the topics addressed in our work.

In Section 2.1, we introduce the foundational concepts of FL, emphasizing its decentralized training architecture and privacy-preserving model aggregation scheme. We detail the typical workflow of FL and highlight the essential hyperparameters that govern the training process, including the number of rounds, client participation rates, batch size, and learning rate. Furthermore, we distinguish between static and dynamic client selection strategies and explain how deployment strategies; specifically Horizontal Federated Learning (HFL) and Vertical Federated Learning (VFL); influence system design depending on data alignment across clients. In Section 2.2, we examine the critical challenge of data heterogeneity, formally referred to as non-Identically and Independently Distributed (non-IID) data. We classify non-IIDness into several types, such as label skew, feature imbalance, quantity skew, and temporal drift, and present their implications on model convergence. Section 2.3 introduces Hedonic Coalitional Games, a class of cooperative games where agents form coalitions based on preference relations. We discuss their formulation, preference structures, and stability concepts, and explain their relevance for modeling client grouping and coalition formation in FL environments. In Section 2.4, we present Stackelberg Games, which model hierarchical leader-follower dynamics. We describe their mathematical structure and demonstrate how they can be leveraged in FL for designing layered incentive systems, particularly when resourceful and less-capable clients must coordinate through asymmetric roles. Section 2.5 covers Reverse Auction Games; a mechanism for decentralized resource procurement, and highlights how they are employed in FL marketplaces to enable clients to competitively bid for training tasks, ensuring compensation while optimizing participation and cost efficiency for the task owner. Section 2.6 focuses on the Dirichlet distribution, a common technique used to simulate synthetic non-IID data across clients. We formalize the distribution, discuss its use in FL experimentation, and critically analyze the limitations of the traditional single-level Dirichlet approach; specifically its inability to capture realistic feature or quantity skew. Finally, in Section 2.7, we present a comprehensive review of related works, focusing on different key areas, such as: non-IID data mitigation strategies, detection and handling of selfish clients, intelligent client selection mechanisms, the use of Stackelberg games in federated settings, and incentive-driven frameworks including reverse auctions and marketplace-based designs.

2.1 Federated Learning

In its core, FL [1] is a distributed mechanism in which the training happens without the need to offload collections of data towards a central repository, instead, end users, i.e., clients, perform the training on their devices using their own data, and create a set of local models which are sent to a federated server that aggregates them into a global model. This global model is shared with the clients for more training iterations for improvement. Using such an approach, devices would benefit from the data of the other devices without actually having access to them.

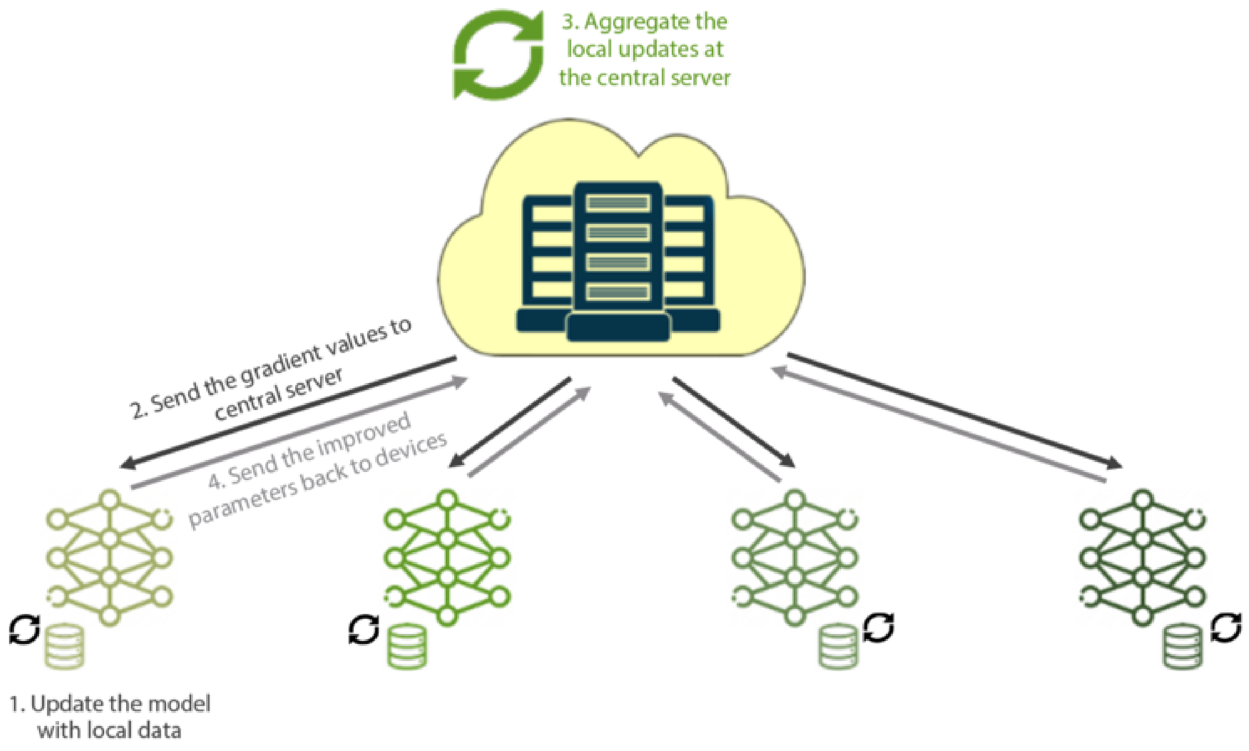

3. Aggregate the local updates at the central server
2. Send the gradient values to central server
4. Send the improved parameters back to devices
1. Update the model with local data

Figure 2.1 Federated Learning Flow

Below we summarize the general steps towards accomplishing FL, where at time=0, the central server has an initial global model. The server deploys this global model to the clients, which have the same model at this phase.

1. Each client trains its local data on this model and comes out with a new local model.
2. The clients send their local models to the central server.
3. The central server aggregates the local models, to form a new global model.

4. The central server sends the aggregated global model back to the devices.

The processes gets repeated until a desired accuracy convergence is established, or a specific number of rounds take place.

2.1.1 Hyperparameters

In FL, a set of hyperparameters must be defined to govern the training process. These parameters directly influence the performance, efficiency, and scalability of the FL system and are often fine-tuned based on the training task, underlying hardware capabilities and system constraints of participating devices. Key factors influencing parameter selection include computational power, memory availability, and communication bandwidth. These parameters are typically adjusted iteratively during the experimentation phase to achieve an optimal trade-off between accuracy, convergence speed, and system cost.

Below, we outline and discuss the key parameters in a typical FL setup:

1. **Number of Federated Rounds (T):** Defines the total number of communication rounds. Higher values improve accuracy but increase latency and communication cost.
2. **Total Number of Clients (K):** Represents the full pool of potential participants in the FL process. In real-world scenarios, this may range from a few dozen to thousands. Larger K increases diversity but may require sophisticated selection or sampling strategies.
3. **Fraction of Participating Clients per Round (C):** Determines the fraction of clients sampled each round: $m = \lceil C \cdot K \rceil$. Lower C reduces overhead but risks statistical bias. Client selection can be:
   - **Static:** Fixed client set throughout training.
   - **Dynamic:** Clients change per round.
4. **Local Batch Size (B):** Specifies the number of data samples used in a single local update during local training. Common values are 64, 128, 256, and 512.
5. **Number of Local Epochs (E):** Indicates how many passes each client makes over their local dataset per round. Higher E accelerates global convergence, but exacerbate model divergence, especially under non-IID data, while being computationally consuming.

6. **Learning Rate (η):** Governs the step size during gradient descent. It is crucial for stability and convergence.

The interplay between these hyper parameters significantly impacts the convergence behavior and fairness of federated learning systems. Adaptive tuning strategies, such as learning rate decay, or adaptive aggregation weights, can further improve efficiency in dynamic and heterogeneous environments.

2.1.2 Deployment Strategies

FL can be deployed under different structural paradigms depending on the nature of data partitioning across participating entities. These paradigms are primarily classified into Horizontal Federated Learning (HFL) and Vertical Federated Learning (VFL) [2], based on the overlap of features and data samples across clients.

1. **Horizontal Federated Learning (HFL):** Also referred to as sample-partitioned FL, this strategy applies when participating clients (e.g., devices, institutions, or companies) hold datasets that share the same feature space but differ in their sample space. In other words, all clients collect the same types of information (i.e., the same set of features), but on different subsets of the population.

   Example: Consider two competing email service providers operating in the same region. While their customer bases are largely disjoint, both providers collect similar features such as user ID, email frequency, attachment size, and storage usage. In this scenario, the underlying feature schema is aligned across clients, but the datasets correspond to different individuals. HFL enables the training of a shared model (e.g., for spam detection or quota prediction) without transferring raw data between providers.

2. **Vertical Federated Learning (VFL):** Also known as feature-partitioned FL, this strategy is appropriate when clients possess datasets with overlapping sample spaces but different feature sets. In this setup, multiple entities hold complementary information about the same population of users, and wish to jointly train a model by aligning their features in a privacy-preserving manner.

   Example: Consider a conglomerate that offers both medical insurance and life insurance services. A large portion of its customers may subscribe to both offerings, but the internal databases of the two divisions store different features. The medical insurance branch may record health records, diagnosis history, and hospital visits, while the life insurance branch stores income, risk preferences, and coverage amounts. VFL allows these departments to collaboratively train a predictive model (e.g., for cross-sell targeting or risk scoring) by leveraging the union of feature spaces without sharing raw data.

Furthermore, based on the nature of the participating client devices, federated learning deployments can be broadly categorized into two main types.

1. **Cross-Silo Federated Learning:** Cross-silo FL refers to deployments where the participating entities are typically organizations, institutions, or data centers. The number of clients is relatively small (e.g., tens to hundreds), and each participant has substantial computing power and large datasets. The communication is more stable and participation is consistent across rounds. Cross-silo FL is common in industries such as healthcare, finance, or telecommunications, where institutions collaborate while respecting privacy regulations.

   Example: Several hospitals jointly train a diagnostic model without sharing patient data, each hospital acting as one silo.

2. **Cross-Device Federated Learning:** Cross-device FL involves training across a very large number of edge devices (e.g., smartphones, IoT sensors). In this setting, each device holds only a small amount of data, and system heterogeneity (limited bandwidth, power, or availability) has a large impact on the final global model output. Cross-device FL is widely used in applications where the underlying data that will fuel the training is personal, and generated by the owners of these edge devices.

   Example: A mobile service provider trains a predictive text model across thousands of smartphones, with each device contributing locally observed typing patterns.

Understanding both the partitioning paradigms (HFL and VFL) and the deployment types (cross-silo and cross-device) is essential for selecting the appropriate federated learning framework. These classifications jointly determine the communication protocols, privacy requirements, and client selection strategies needed in practice.

2.2 Non-Identically and Independently Distributed Data (non-IID)

One of the most pressing challenges in FL is the presence of non-Identically and Independently Distributed (non-IID) data across participating clients [7]. Unlike traditional machine learning paradigms, FL systems must contend with data that vary significantly in terms of quantity, distribution, and semantics across devices. This heterogeneity arises from the fact that each client collects and generates data in different contexts; driven by user behavior, hardware characteristics, geographic location, or application-specific usage patterns. The presence of non-IID data has been shown to degrade the performance of FL process in regards of the global model's accuracy, leading to slower convergence, biased, and reduced generalization capabilities [12]. For example, the authors of [13] study the disadvantages of non-IID data in FL environments by conducting extensive experiments. They reiterate the fact that despite state-of-the-art approaches which try mitigating the downsides of such heterogeneity, non-IID data still causes a challenge for the model accuracy convergence in FL.

Hereafter, we categorizes Non-IIDness in FL into several distinct types, each affecting the training process in different ways, and we highlight research works that have proposed methods of mitigations:

1. **Class Imbalance:** Also known as, label distribution skew, occurs when the distribution of labels across clients is not uniform. In this setting, some clients may only possess data samples belonging to a limited subset of classes, or may have heavily imbalanced class proportions. For instance, in an image classification task using the CIFAR-10 dataset, one client may predominantly have samples from the 'cat' and 'dog' classes, while another may only have 'airplane' and 'automobile' samples. This causes local model updates to overfit to the prevalent local classes and limits the global model's ability to generalize across the full label space. Label distribution skew is one of the most studied forms of non-IIDness in FL, and synthetic simulations often use the Dirichlet distribution to control its severity.

2. **Feature Imbalance:** This kind of imbalance arises when the distribution of input features differs across clients, even if the label distribution remains the same. For example, in a handwriting recognition task, different clients may use devices with varying screen resolutions, or their users may have different writing styles. Although the class labels (e.g., digits 0–9) are shared across clients, the underlying features used to represent those classes vary significantly. This heterogeneity impacts the learning process by introducing mismatches between the client-specific data manifold and the global feature space, thereby impeding convergence and resulting in poor performance on unseen feature distributions.

3. **Quantity Skew**: This refers to the unequal number of training samples held by each client. This is often a result of natural variations in user engagement, device usage,

or data collection capabilities. In real-world deployments, some clients may contribute tens of thousands of training samples, while others may only have a few dozen. This imbalance causes certain clients to dominate the model update process when weightings are based on local data sizes, leading to biased global models. Moreover, clients with smaller datasets may not contribute meaningfully to the learning process, exacerbating the disparity in representation and increasing the risk of underfitting minority populations.

4. **Concept or Label Shift**: This refers to a scenario where the relationship between input features and output labels varies across clients. For instance, two hospitals may record similar patient vital signs, but differences in diagnosis protocols may cause the same feature vector to map to different labels. In this case, local models may learn conflicting mappings, which destabilizes global model aggregation. This type of heterogeneity is particularly challenging because it implies that the underlying labeling function is not stationary across clients, invalidating the assumption of a shared learning objective.

5. **Semantic Skew**: This occurs when identical labels represent semantically different concepts across clients. For example, the label 'home' in a smart assistant dataset might refer to a physical location for one user, while representing a web page or app interface for another. This type of skew introduces ambiguity in the learning process, as the same class label is associated with disparate feature distributions and meanings across clients. Semantic skew can be seen as a combination of label and feature skew, and its resolution often requires personalized model architectures or client-specific adaptations.

Given the heterogeneous nature of client devices in FL, data generated by these devices will also be heterogeneous (i.e., non-IID). This plays against the training of FL models as it causes delays in the desired accuracy convergence of the global model. Several works in the literature have addressed this challenge through different strategies. For example, [14] survey a range of techniques such as data sharing, data augmentation, local fine-tuning, personalization layers, and multi-task learning. Other approaches have focused more specifically on model adaptation: [15] proposes a server-side knowledge-distillation method to align local and global models, while [16] introduces a unified feature learning framework combining adversarial modules with consensus losses to reduce representation and optimization inconsistencies. In addition, the work in [17] provides a detailed experimental analysis of how non-IIDness affects different layers in deep models, showing that post-calibrating the classifier can improve classification performance.

2.3 Hedonic Coalitional Games

Coalitional Hedonic Games (CHGs) are a subclass of cooperative game theory in which players form coalitions based on their preferences over the group of players they are associated with. In other words, in CHGs, the utility of a player depends entirely on the identity of the members of their coalition, not on any external outcomes or allocations. These games are particularly well-suited for modeling scenarios where agents self-organize into groups based on subjective preferences, such as social networks, peer-to-peer systems, or client communities in FL environments.

2.3.1 Formal Definition

Let $\mathcal{N} = \{1, 2, \ldots, n\}$ be a finite set of players. A *coalition* is any subset $C \subseteq \mathcal{N}$ such that $i \in C$ for a player $i \in \mathcal{N}$. A *coalition structure* (or *partition*) π is a partition of $\mathcal{N}$ into disjoint coalitions, i.e., $\pi = \{C_1, C_2, \ldots, C_k\}$ such that $\bigcup_{j=1}^{k} C_j = \mathcal{N}$ and $C_j \cap C_{j'} = \emptyset$ for all $j \neq j'$.

Each player $i \in \mathcal{N}$ has a preference relation $\succeq_i$ defined over the set of coalitions that include i. That is, for two coalitions C and D such that $i \in C$ and $i \in D$, we write $C \succeq_i D$ to indicate that player i weakly prefers being in coalition C over coalition D.

A *Hedonic Game* is defined by the tuple $(\mathcal{N}, \{\succeq_i\}_{i \in \mathcal{N}})$.

2.3.2 Types of Preferences

Several classes of preference structures are commonly studied in CHGs:

- **Top Responsive Preferences**: Each player has a fixed set of favorite players (friends). A player prefers coalitions that contain more of their favorite players.
- **Bottom Avoiding Preferences**: Each player has a set of disliked players (enemies). A player prefers coalitions that contain fewer of their enemies.
- **Additively Separable Preferences**: Each player assigns a value $v_i(j)$ to every other player $j \neq i$, and the utility of a coalition C for player i is the sum of values over members in C:

  $$u_i(C) = \sum_{j \in C \setminus \{i\}} v_i(j)$$

  In this case, $C \succeq_i D$ if and only if $u_i(C) \geq u_i(D)$.

Additively separable preferences are particularly useful in engineering systems, such as FL, where the utility of a client may depend on the characteristics of its coalition members (e.g., data quality, communication bandwidth, or computational resources).

2.3.3 Stability Concepts

An important focus in CHGs is the stability of coalition structures. A partition π is said to be stable if no player has an incentive to move to a different coalition or to form a new one. Common notions of stability include:

- **Nash Stability (NS)**: A partition π is Nash stable if no player i would prefer to unilaterally move to another coalition (or form a singleton) while being accepted by that coalition:
$$\nexists C \in \pi \cup \{\emptyset\} \text{ such that } C \cup \{i\} \succ_i \pi(i)$$
where $\pi(i)$ is the coalition containing i in π.

- **Individual Stability (IS)**: A partition is individually stable if no player can move to another coalition such that they strictly prefer it and no one in the new coalition objects:
$$\nexists C \in \pi \cup \{\emptyset\} \text{ such that } C \cup \{i\} \succ_i \pi(i) \text{ and } \forall j \in C, C \cup \{i\} \succeq_j C$$

2.3.4 Coalitional Hedonic Games Application to FL

In FL systems, coalitional games are a natural fit for modeling the formation of client coalitions based on system compatibility, or shared objectives. For instance, clients may prefer to participate in learning tasks with others whose data distributions are similar (to improve convergence) or whose resource capabilities are compatible (to avoid stragglers). The preference structure can be defined via similarity functions or cooperative utility functions that capture these objectives.

Accordingly, coalitional Hedonic Games provide a flexible and powerful framework for modeling decentralized, preference-driven group formation in multi-agent systems, such as a federated learning environment. Their abstraction aligns well with client-side autonomy in FL, where clients may benefit from forming coalitions that optimize for compatibility and shared interests. By leveraging CHGs, self-organizing federated systems can be designed that promote stability, fairness, and collaborative efficiency without requiring centralized control. In

Chapter 3, we apply this foundation by developing a trust-aware client selection framework that integrates hedonic and coalitional game models.

2.4 Stackelberg Games

Stackelberg games are a class of hierarchical game-theoretic models that capture the strategic interaction between two groups of players: leaders and followers. The defining characteristic of Stackelberg games is the sequential nature of decision-making. The leader commits to a strategy first, and the followers, upon observing the leader's decision, choose their strategies accordingly to maximize their own utilities. This framework is well-suited for modeling scenarios involving power asymmetry, and decision hierarchies in distributed systems.

2.4.1 Formal Definition

Consider a two-player Stackelberg game involving a leader L and a follower F. Let $x \in X$ denote the strategy chosen by the leader, and $y \in Y$ the strategy chosen by the follower. The leader and follower have utility functions $U_L(x, y)$ and $U_F(x, y)$, respectively.

The Stackelberg game is formulated as a bilevel optimization problem:

$$\max_{x \in X} \; U_L(x, y^*(x)) \quad \text{subject to} \quad y^*(x) \in \arg\max_{y \in Y} U_F(x, y)$$

Here, $y^*(x)$ is the best response of the follower to the leader's choice x. The leader anticipates the follower's reaction and selects x accordingly to optimize its own objective.

2.4.2 Stackelberg Equilibrium

A pair (x^*, y^*) constitutes a Stackelberg Equilibrium if:

$$\begin{aligned} y^* &\in \arg\max_{y \in Y} U_F(x^*, y) \\ x^* &\in \arg\max_{x \in X} U_L(x, y^*(x)) \end{aligned}$$

That is, x^* is optimal for the leader, assuming the follower responds with y^*, which in turn is the follower's best response to x^*.

In simultaneous-move games, players choose their strategies at the same time, without observing the others' choices. By contrast, Stackelberg games are sequential: the leader commits to a strategy first, and the follower then selects a best response after observing the leader's action. This sequential structure refines the equilibrium concept, as the leader anticipates the follower's reaction when optimizing its own decision.

2.4.3 Stackelberg Games Application to FL

In the context of FL, Stackelberg games provide a powerful framework to model hierarchical interactions between entities such as the network owner (or the FL process orchestrator, or the model aggregator) and the participating clients, or between a resourceful client and other less-resourceful clients. Stackelberg games offer a hierarchical and anticipatory model of interaction that is well-suited to FL environments where asymmetry exists between the coordinating leader and participating followers. In Chapter 4, we build on this foundation by introducing a framework where resourceful clients act as leaders forming coalitions with less-resourceful followers. This leader–follower structure democratizes participation, balances heterogeneity, and provides a practical instantiation of Stackelberg games tailored to federated learning environments.

2.5 Reverse Auction Games

Reverse auction games are a class of auction mechanisms where the roles of buyers and sellers are inverted compared to traditional auctions. In a reverse auction, a single buyer (or a central authority) solicits bids from multiple sellers (or service providers), and typically selects the lowest (or most favorable) bid that meets its requirements. Reverse auctions are particularly effective in resource procurement, outsourcing, and service marketplaces, and can be useful in distributed systems like FL, where the central aggregator needs to procure services from a pool of clients.

2.5.1 Formal Definition

Let $\mathcal{N} = \{1, 2, \ldots, n\}$ be a set of n participants (sellers or service providers). A buyer requires a task to be executed or a resource to be provided. Each seller $i \in \mathcal{N}$ submits a bid b_i indicating the price at which they are willing to sell, or to provide the service. The buyer selects one or more winners based on these bids and allocates payments accordingly.

Let:

- c_i be the private cost of agent i to sell, or execute the task.
- b_i be the bid submitted by agent i (not necessarily equal to c_i).

- $x_i \in \{0,1\}$ be a binary variable indicating whether agent i is selected ($x_i = 1$) or not ($x_i = 0$).
- p_i be the payment given to agent i if selected.

The buyer solves the following optimization problem:

$$\min_{\mathbf{x}} \sum_{i=1}^{n} b_i x_i \quad \text{subject to constraints (e.g., budget, quantity, diversity)}$$

The utility of each participant i is defined as:

$$U_i = \begin{cases} p_i - c_i & \text{if } x_i = 1 \\ 0 & \text{if } x_i = 0 \end{cases}$$

2.5.2 Reverse Auctions in FL Marketplaces

In an FL marketplace, a Network Owner (NO), who is commisioned by a task owner for a given learning task and objective, seeks to train a model using privately-held data distributed across clients in the FL system. Instead of commanding participation, the NO can issue a task as a reverse auction where clients bid to offer their resources for a proposed price.

- The buyer is the Network Owner that is wanting to acquire training services.
- The sellers are the clients who bid based on their costs to perform the local training and data sharing.

The network owner evaluates the bids and selects a subset of clients that optimize for cost, data diversity, quality, or performance, subject to system constraints (e.g., latency, client availability). Therefore, using reverse auctions in FL marketplaces offers several advantages:

- **Incentive Compatibility**: With proper design, reverse auctions encourage truthful bidding and fair compensation.
- **Client Autonomy**: Clients retain control over participation, aligning with the decentralized nature of FL.
- **Cost Efficiency**: The network owner can minimize total expenditures while meeting performance objectives.

Accordingly, reverse auction games offer a powerful mechanism for decentralized resource allocation in FL marketplaces. By allowing clients to bid competitively for participation, these auctions promote fair compensation, preserve autonomy, and optimize task owner objectives. When coupled with incentive mechanisms and quality-aware selection strategies, reverse auctions serve as a foundational tool for designing scalable, fair, and economically efficient FL ecosystems. In Chapter 5, we formalize the structure of the FL marketplace. In particular, we differentiate the roles of the Task Owner and the Network Owner, introducing the latter as a distinct entity responsible for coordinating participation and enforcing incentive alignment, thereby achieving a clearer separation of responsibilities and governance in the system.

2.6 Dirichlet Distribution

The Dirichlet distribution is a multivariate generalization of the Beta distribution and is widely used in Bayesian statistics and probabilistic modeling. In the context of FL, the Dirichlet distribution has become a well-used technique for simulating heterogeneous, non-IID data among clients. It allows for controllable label skewness by adjusting a concentration parameter, making it a convenient tool to benchmark FL algorithms under various degrees of non-IID conditions.

2.6.1 Mathematical Formulation

Let K be the number of classes in a classification task. The Dirichlet distribution defines a probability vector $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_K)$ such that:

$$\theta_i \geq 0 \quad \text{for all } i = 1, \ldots, K, \text{and} \sum_{i=1}^{K} \theta_i = 1$$

The Dirichlet distribution with parameter vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_K)$ has a probability density function:

$$\mathrm{Dir}(\boldsymbol{\theta}|\boldsymbol{\alpha}) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{i=1}^{K} \theta_i^{\alpha_i - 1}$$

where $B(\boldsymbol{\alpha})$ is the multivariate Beta function:

$$B(\boldsymbol{\alpha}) = \frac{\prod_{i=1}^{K} \Gamma(\alpha_i)}{\Gamma\left(\sum_{i=1}^{K} \alpha_i\right)}$$

and $\Gamma(\cdot)$ denotes the Gamma function. The vector $\boldsymbol{\alpha}$ controls the concentration of the distribution:

- If all $\alpha_i = 1$, the distribution is uniform over the simplex.
- If all $\alpha_i < 1$, the distribution becomes sparse, assigning most probability mass to few classes.
- If all $\alpha_i > 1$, the distribution becomes more balanced.

A commonly used simplification is the *symmetric Dirichlet distribution*, where $\alpha_i = \alpha$ for all i. The single parameter α then controls the degree of heterogeneity: lower values of α create more imbalanced and non-IID data partitions.

2.6.2 Dirichlet Distribution in FL

To simulate non-IID data in FL, researchers often apply the Dirichlet distribution to control the label distribution of data across clients. For each class k, a proportion vector $\boldsymbol{\theta}^{(k)}$ is drawn from $\mathrm{Dir}(\alpha)$, where $\theta_i^{(k)}$ indicates the fraction of class k's data assigned to client i. By repeating this process across all classes, distributions of datasets are generated where clients receive differing numbers and combinations of labels.

This method allows for a controlled and reproducible way to generate non-IID partitions from a shared dataset. For instance:

- $\alpha = 0.5$ produces highly skewed distributions with strong client-label specialization.
- $\alpha = 10$ yields more IID distributions with balanced class proportions.

While the single-level Dirichlet distribution provides a convenient and widely adopted mechanism for simulating non-IID client data, its limitations, such absence of taking the feature skewness into account, and lack of subpopulation modeling, restrict its ability to fully capture the complexities of real-world heterogeneity that inherently present in FL environments. In Chapter 6, we address these shortcomings by introducing a two-level Dirichlet partitioning framework that extends the traditional model to incorporate both feature-level and label-level variations, enabling more realistic benchmarking of FL systems.

2.7 Literature Review

In this section, we survey the literature and state-of-the-art related to our work. We begin by examining how data heterogeneity destabilizes optimization and fairness, and how recent methods attempt to diagnose or mitigate these effects. We then turn to the role of selfish or free-riding clients, whose behavior degrades global utility and erodes participation incentives, motivating mechanisms for detection, exclusion, and incentive alignment. Building on these foundations, we review client-selection policies that trade off statistical benefit, system constraints, and participation dynamics, followed by game-theoretic approaches. Finally, we review works on incentives, reputation, and data valuation, highlighting how economic incentives and reputation updates interact.

2.7.1 Challenges of Data Heterogeneity in FL

Non-independent and identically distributed (non-IID) data remains one of the primary challenges in FL, as it often degrades model convergence and. A substantial body of research has sought to characterize and address non-IID conditions, each offering valuable insights.

The authors of [7] provide a comprehensive overview of FL and discuss the challenges posed by data heterogeneity across client devices. This includes how non-IID data can lead to discrepancies in local model updates, which, when aggregated, may result in suboptimal global models. They categorize various strategies to address data heterogeneity, including approaches like data sharing among clients, clustering of clients with similar data distributions, and personalized model training to accommodate diverse data characteristics. In [6], the authors highlight the necessity of moving beyond simplistic non-IID generation methods and emphasize that realistic FL benchmarks should capture more nuanced forms of data heterogeneity. They discuss how non-IID data is not just about imbalanced class distributions but also includes variations in data distributions due to differences in user behavior and domain shifts. Their work calls for better evaluation methodologies that account for these real-world complexities rather than relying on artificial partitioning strategies commonly used in FL research.

The authors of [18], [13], [19], and [20] highlight the impact of heterogeneous data on the global model. In [18], the authors explore the significant challenges posed by non-IID data in FL, and demonstrate that when data across client devices is highly skewed; such as when each client predominantly holds samples from a single class; model performance can deteriorate by up to 55%. In [13], the authors also address the challenges posed by non-IID data in FL. To aid researchers in understanding and studying non-IID settings, they evaluate the performance of FL algorithms under diverse non-IID conditions, providing insights into how data heterogeneity impacts model convergence and accuracy. The authors of [19] discuss the critical challenge of data heterogeneity in FL environments. The authors propose a methodology to evaluate the level of compatibility between clients. This assessment aims to identify the degree to which local datasets deviate from the IID assumption, which is often violated in practical FL scenarios. By quantifying this compatibility, the study provides insights into the impact of non-IID data on the performance and convergence of FL models. The work in [20] investigates the impact of neural network architectures on FL performance, particularly under conditions of data heterogeneity. The authors conduct a comprehensive empirical study comparing self-attention-based architectures, such as Vision Transformers (ViTs), with traditional convolutional neural networks (CNNs) across various FL algorithms and datasets. Their findings reveal that ViTs exhibit greater robustness to distribution shifts inherent in non-IID data, leading to reduced forgetting, accelerated convergence, and improved global model performance. The work in [21] explicitly acknowledges that real-world FL data heterogeneity arises not just from class imbalance but also from feature-level divergence across clients. Their experiments reveal that accounting for structured feature variations yields significantly better performance in highly non-IID settings. [22] further emphasizes that client-specific data characteristics highlights the limitations of global modeling under both feature and label divergence.

On the other hand, works such as [23], [24], [25], [26] and [27] showcase the negative effects of data heterogeneity in FL, and propose possible solutions to dampen their effect on the global model using variety of techniques. In [25] the authors propose Clustered Federated Learning (CFL), a framework that groups clients into clusters based on the similarity of their data distributions. Within each cluster, a shared model is trained, allowing the framework to handle non-IID data more effectively. Their approach is model-agnostic and operates under privacy constraints, ensuring that client data remains decentralized and secure. The paper demonstrates that CFL can achieve better performance compared to traditional FL methods, particularly in scenarios with significant data heterogeneity. The authors of [27] address the challenge of data heterogeneity in FL by proposing a novel approach called Federated Feature Distillation (FedFed), which partitions data features into two categories: performance-sensitive features that significantly contribute to model performance and performance-robust features with limited impact. In FedFed, performance-sensitive features are shared globally among clients to mitigate data heterogeneity, while performance-robust features remain local to preserve privacy. This strategy enables clients to train models using both local and shared data, balancing the trade-off between privacy and performance.

The work in [23] investigates how non-IIDness among clients impact FL performance and proposes a method to synthesize datasets with varying degrees of identicalness using Dirichlet distribution to simulate different levels of data heterogeneity across clients. They introduce a server momentum technique, which adjusts the global model update by incorporating a momentum term to counteract the adverse effects of non-IID data. Meanwhile, [24] investigates the data heterogeneity problem in models utilizing batch normalization, where the degree of data skew directly correlates with the severity of accuracy loss. To address this challenge, the authors introduce SkewScout, a system-level approach that dynamically adjusts the communication frequency of decentralized learning algorithms based on the accuracy loss induced by data skew between partitions. The work in [26] also addresses the challenges posed by non-IID data in FL, and propose a method called Federated Representation Augmentation FRAug which focuses on augmenting the feature representations to enhance the model's robustness against non-IID data distributions. The approach involves generating diverse feature representations during training, which helps in aligning the feature distributions across different clients.

The authors of [28] propose Bayesian FL framework called FedNP, that directly addresses the challenge of training on non-IID data by estimating and incorporating a latent global data distribution into the local training process. Unlike conventional FL approaches that suffer from model drift due to biased local data, their approach uses an auxiliary probabilistic task to prevent local models from overfitting to their own distributions. The authors of [29] address data heterogeneity by generating virtual datasets that mimic diverse client distributions without sharing raw data. The approach employs meta-learning to create synthetic feature-label pairs, enabling clients to preemptively adapt to unseen distributions during training. The authors of [30] focus on classifier-layer biases in non-IID settings. Their study analyzes how skewed label distributions propagate errors to deeper network layers. In [31], the authors propose a method that decomposes convolutional filters into shared "basis atoms" and client-specific coefficients. By disentangling common and unique features, it reduces variance in model updates while preserving client-specific patterns. In [32], the authors investigate non-IID challenges in multi-modal FL and highlight performance degradation when clients have mismatched modality distributions. It proposes modality alignment techniques using contrastive learning to harmonize representations across different data streams.

On the other hand, several works have proposed algorithmic solutions to mitigate the impact of data heterogeneity. For example, pFedMe [33] addresses personalization through bi-level optimization using Moreau envelopes, decoupling local model adaptation from global model training and achieving improved convergence under heterogeneous client data distributions. The authors in [34] propose an entropy-based data selection mechanism that filters out low-quality, high-entropy samples on each client to improve training efficiency in federated learning. This lightweight, client-side approach reduces computational cost and maintains model accuracy under non-IID settings. The work in [35] introduced a novel method that uses diffusion models on clients' local data to generate synthetic images, which are then used to retrain the final classification layer of the global model. The approach boosts accuracy and convergence in non-IID settings by mitigating classifier divergence. Similarly, the work in [36] proposes a diffusion generation based method that synthesizes local data on clients to alleviate the effects of non-IID data in tasks. By balancing class distributions through synthetic data, it enhances the robustness and convergence of federated models. The work in [37] discusses a feature-matching based data synthesis approach that generates auxiliary data on clients to match global feature distributions and reduce the impact of non-IID data. By training a conditional generator guided by a global feature prototype, it improves convergence and accuracy across FL tasks. The work in [38] tackles non-IID data challenges in FL by relying on a generator-based personalization framework, that shares a feature extractor and learns a generator that can reconstruct synthetic data representations per client to align learning across heterogeneous environments. Similarly, the work in [39] proposes a personalized federated learning framework that generates individualized synthetic data on each client by training a local generator aligned with the client's model and data distribution. It improves local model performance and mitigates data heterogeneity without data sharing.

2.7.2 Free-riders & Selfish Clients in FL

The problem of selfish or free-riding clients is a critical challenge in FL. Unlike honest participants which contribute meaningful updates, free-riders consume system resources without providing proportional value, thereby degrading global model accuracy, wasting communication bandwidth, and undermining fairness. Their presence not only reduces the reliability of the aggregation process but also discourages long-term participation from contributing clients. Consequently, recent research has devoted effort to analyzing free-rider behavior, measuring its impact, and proposing detection or mitigation mechanisms ranging from auditing and clustering to reputation- and similarity-based approaches.

In [40], the authors evaluate and perform a theoretical and experimental analysis pertaining to the presence of selfish clients in FL. In [41], the authors highlight the detrimental effects of selfish clients and their impact on other client's long term participation in the FL rounds. They propose a method to combat this by treating the selfish behavior of the clients as infinitely repeated game and derive the optimal Nash Equilibrium. Simulation results shows that their approach can decrease the number of selfish devices tremendously. In [42], the authors propose a a defensive mechanism against selfish clients in FL called PASS (Parameter

Audit-based Secure and fair federated learning Scheme). The approach relies on an contribution evaluation where non-contributing clients are left out from the learning rounds if they do not contribute above a certain threshold. In [43], the authors propose a framework that analyzes the behavior of clients in FL. They highlight the importance of sharing of data, as well as the negative impact of selfish clients in the learning rounds. Based on their findings, they propose an accuracy-shaping based mechanism which maximizes the data generation, while decreasing the selfishness of the clients in the environment.

The authors in [44] perform a feasibility study that uses several statistical metrics to measure similarity between clients and cluster them accordingly. Then, a single client from each cluster is used to for each round to participate in the learning. This is to promote the dissimilarity of selected clients that contribute differently to each round. In [45], the authors propose a modification of the FL process by introducing a hierarchical clustering step that aims to separate the clusters of clients by using the similarity of their local updates against the global joint model. The authors use a couple of similarity distance metrics, and once the clusters formed, the training takes place independently and in parallel on specialised models in different clusters. Similarly, in [46], the authors propose a clustering approach of different clients based on the similarity distance between their models and perform the training on each cluster with a different model. On the other hand, the authors of [47] highlight the impact of free-riders on the accuracy of the global model using empirical experiments, and propose the usage of euclidean distance as a similarity metric to detect free-riders. In [48], the authors propose a novel free-rider mechanism detection in the realm of FL that relies on Deep Auto-encoding Gaussian Mixture Model (DAGMM). DAGMM utilizes a deep Auto-encoder and generates a low dimensional representation of the model by calculating the reconstruction error for all input. The approach also concatenates the low-dimensional embedding with the distances between the input and the output using distance metrics such as Cosine and Euclidean. Then, the output obtained is inserted to a Gaussian mixture model (GMM) that takes an estimation network while training the parameters and make the input fit the GMM. The authors also extend their approach and propose STD-DAGMM which is more robust. They achieve this by incorporating the standard deviation metric into DAGMM network through appending it to the input vector if the estimation network. The authors of [49] propose a new FL mechanism called FRAD that uses a mixture of contribution values, reputation based approaches, as well as DAGMM from [48] to detect possible free-riders. The way FRAD works is, based on the computing resource, communication cost, and data quality, initially the authors model the contribution values for each device and use the PageRank based algorithm to create a reputation-based model that is able to select benign clients in a fair manner. This is enforced with a deep auto-encoding Gaussian mixture model

(DAGMM) that relies on a combination of euclidean distance and cosine similarity, that combines historical contribution with reputation scores for each device to detect remaining free-riders.

2.7.3 Client Selection Mechanisms in FL

Effective client selection is critical to the success of federated learning, directly impacting model performance, training efficiency, and system robustness. We survey works that propose strategies to optimize client participation under various system and data heterogeneity settings.

The authors of [50] use the devices' geographical locations and resource capabilities to derive a list of devices that can handle the tasks at a given FL round. They treat this as a maximization problem and work on a solution using heuristic approaches in which the server tries to maximize the number of devices per round. The authors of [51] propose a scheduling approach between the federated server and clients by taking into account the trust factor between them. To do this, they use the resource capabilities of the devices coupled with a Double Deep Q Learning (DDQN) based scheduling approach. The authors of [52] also investigate the problem of client selection inside wireless networks. They base their approach on prioritizing the clients that would have a more accurate models for the overall global model. Once clients are selected, they are given resource blocks (RB) to communicate with the federated server. The authors also make use of Artificial Neural Networks (ANN), especially with devices not selected for the learning rounds as they try to extract possible gradients from them. These gradients further help coming up with better global models.

In [53], the authors highlight client selection in live and dynamic environments as they propose an on-demand client selection scheme. Their approach relies on containerization technologies to construct the environments as they tackle the problem of availability, especially in FL areas which lack enough available clients with adequate resources needed to participate in the learning rounds. In [54], the authors base their client selection mechanism on the data found at each client. They argue that the varying degrees of non-IID data at each client play a big role in the accuracy degradation of the global model. Therefore, they use weight divergence to estimate the non-IID degree of the clients. They accordingly propose a new FL algorithm called *CSFedAvg* where clients that have lower degrees of non-IID data have a preference by the federated server to get selected to train models.

The authors of [55] introduce an Active Client Selection for Clustered Federated Learning (ACFL), a novel approach that operates under a clustered FL (CFL) framework. Instead of relying on random or full client participation, ACFL employs active learning metrics;

such as uncertainty sampling, query-by-committee (QBC), entropy, and loss-based strategies; to select the most informative clients within each cluster. The authors of [56] introduce Federated Prototype Rectification with Personalization (FedPRP), a framework designed to address skewed class distributions and personalization simultaneously in FL. Their proposed setting, called Skewed Heterogeneous Federated Learning (SHFL), reflects practical data distributions where certain classes dominate, and others are underrepresented across clients.

2.7.4 Stackelberg Games in FL

In this section, we review the state-of-the-art client selection approaches in FL, placing particular emphasis on those that employ multi-stage, non-hierarchical Stackelberg games client selection approach.

In [57], the authors provide a comprehensive taxonomy of FL incentive mechanisms, classifying approaches different groups, namely: Shapley value, contract theory, Stackelberg games, and auctions. They highlight limitations in existing strategies, especially in cross-client fairness and adaptive compensation. Complementing this, the authors in [58] stress the importance of incentive robustness and the difficulty of simultaneously achieving fairness, budget constraints, and truthful reporting in dynamic systems.

In [59], the authors introduce a framework that relies on Stackelberg games to enhance convergence in FL over wireless networks. The main focus of the work is on optimizing both the global model's accuracy and communication latency under energy constraints, which are critical in wireless FL environments. The approach separates the learning process into two levels: the server (leader) minimizes global loss by selecting devices with higher-quality data and more recent updates, while the devices (followers) aim to minimize latency through optimal sub-channel assignment and resource allocation. The authors of [60], discuss the implementation of an FL environment at the edge, focusing on efficient resource management and motivation for user participation. They propose an incentive mechanism based on a Stackelberg game where the base station (BS) incentivizes user devices to participate in the learning rounds. The BS (as the game leader) provides rewards to encourage devices to allocate optimal local resources for training, which helps improve global model accuracy. Devices, as followers, choose optimal CPU frequency to maximize their benefits within the constraints of energy and communication costs. The authors of [61] discuss and propose a Stackelberg game-based framework for resource allocation and pricing in mobile edge computing (MEC). In their approach, the MECs act as leaders setting prices for resources, while end-users (devices) act as followers who decide on their resource demands based on these prices and their budget constraints. This approach addresses challenges in distributing limited MEC resources among competing EUs, optimizing both resource allocation and pricing for heterogeneous resources and demands.

On the other hand, in [62], the authors devise an incentive mechanism for an FL system that leverages a two-stage Stackelberg game model. The server (leader) seeks to incentivize workers (followers) to participate in model training by designing utility functions for both parties. In the first stage, the server sets a reward rate, balancing its payment against expected accuracy improvements. In the second stage, workers decide their optimal accuracy levels, considering costs and maximizing their utilities. This game-theoretic approach ensures that the server maximizes its utility while encouraging workers to contribute their local data for the training. The work in [63] addresses the challenge of motivating edge devices to contribute to FL in IoT based applications. They propose a novel incentive mechanism, using deep reinforcement learning and a Stackelberg game framework, to optimize rewards for edge devices that contribute data and computational resources to FL tasks, where the parameter server (leader) sets a reward (pricing strategy), while edge nodes (followers) decide their level of participation based on this reward, with the goal of reaching an equilibrium.

In [64], the authors present a multi-factor incentive mechanism for FL in IoT systems. Their approach, addresses challenges with high costs and low utility due to information asymmetry such as unknown data owner reputation, computation power, and data quantity. Accordingly, they design a two-stage Stackelberg game model, in which the task publisher (leader) sets optimal incentive strategies, and data owners (followers) adjust training strategies for maximum utility. The authors also make use of blockchain to create a secure environment, storing reputation scores and ensuring reliability and traceability in FL tasks. In [65], the authors introduce a framework to manage energy consumption and incentives in FL with mobile edge computing (MEC) systems. The authors propose a Stackelberg game to balance incentives and data usage in FL. MEC servers (leaders) offer incentives based on target accuracy, while mobile devices (MDs, followers) decide data usage for training, considering energy costs. The authors also consider the social welfare and price of anarchy, ensuring that individual incentives align with global performance goals in the framework.

In [66], the authors propose a Stackelberg game where the server, acting as leader, incentivizes mobile devices based on their data volume or accuracy contribution. Two reward policies are explored: size-based and accuracy-based, with deadlines set to encourage efficient training. Similarly, [67] addresses Non-IID data and client drift in Vehicle Edge Computing by introducing a two-stage Stackelberg game. Here, the cloud server leads while roadside units (RSUs) act as followers, optimizing incentives and performance by partitioning data into global and personalized subsets, while preserving historical data to mitigate mobility challenges.

The work in [68] proposes a blockchain-based framework to secure FL with a dynamic incentive mechanism. The approach relies on Stackelberg games, where the task publisher, acting as a leader, sets rewards, while participants (followers) decide their participation. The proposed work uses a dynamic approach that allows rewards to be adjusted based on the client's reputation and bid, ensuring fair compensation, alongside quality contributions. The authors also make use of consortium blockchain, which decentralizes the framework and transparently manages the FL processes, enhancing security against malicious participants. While on the other hand, the work in [69] focuses on creating an effective incentive mechanism in FL for industrial applications. In their approach which employs a Stackelberg game framework, companies (clients) contribute local data and computational resources to train the model while preserving data privacy, while the server acts as the leader, each aiming to maximize their respective payoffs. The goal of the game is to incentivize high-quality clients to contribute significantly while preventing low-resource clients from sending low-quality or fake data, which could compromise the global model's integrity.

2.7.5 Incentives & Reputation Mechanisms in FL

In federated learning, incentives and reputation mechanisms play a central role in ensuring long-term participation, fairness, and trust among heterogeneous and self-interested clients. Since clients bear costs in terms of computation, communication, and privacy risks, they require proper compensation to remain engaged. At the same time, the decentralized nature of FL makes it essential to evaluate and reward clients not only based on their participation, but also on the quality and reliability of their contributions. Incentive mechanisms provide the economic motivation for clients to join, while reputation systems ensure accountability and discourage dishonest or low-quality behaviors. Together, these tools form the essentials of sustainable FL marketplaces and are critical to designing systems that are both technically and economically viable.

The authors in [70] develop a continuous zero-determinant game model that allows servers to shape client strategy convergence. Their approach aims to optimize social welfare in FL by modeling the interaction between the server and devices as a continuous iterative game. The authors of [71] propose an incentive based FL approach that relies on a reputation-based system. The authors model the aggregation process as a mean-field game and design a gradient calculation algorithm based on stochastic differential equations. The authors of [72] propose a contract-theoretic incentive mechanism for FL involving multiple task publishers. Their design enables each task publisher to optimize its profit while offering contracts that satisfy individual rationality and incentive compatibility constraints. The work is effective in modeling publisher–worker interactions and addressing information asymmetry, and focuses on horizontal publisher competition and forward contract issuance. The authors of [73] propose a truthful incentive mechanism for Vertical Federated Learning (VFL) that explicitly models externalities and the impact of one party's participation on another's reward or utility. The system ensures individual rationality and truthfulness via a novel externality-aware utility function and uses linear programming relaxation to solve the winner determination problem efficiently.

On the other hand, the authors of [74] present a Bayesian Stackelberg-game-based incentive mechanism tailored for blockchain-enabled FL in 6G wireless environments. The system models the platform as the leader and clients as Bayesian followers under incomplete information, aiming to minimize free-riding and encourage truthful participation. Incentives are enforced using smart contracts, while a reputation-based penalty mechanism deters dishonest behaviors. In [75], the authors propose a tokenized incentive mechanism that utilizes tokens as rewards for the participating clients. The rewarding mechanism is based on historical accuracy records of the clients alongside random selection to avoid overfitting. In [76], the authors discuss a core-selecting auction mechanism for data sharing in FL. The approach addresses the issue of unfair reward allocation by ensuring group rationality and coalition stability using core solution concepts from cooperative game theory. A matching algorithm is used to form contributor groups, and a minimum-deviation core payment rule ensures truthfulness and efficiency.

The authors of [77] propose a quality-aware incentive mechanism for clients in FL environments. The approach engineers a quality quantification method to assess individual clients' learning quality, while providing an incentive mechanism for their selection. Meanwhile, the work in [78] highlights an incentive mechanism tailored for FL based Recommendation Systems deployed in Mobile Edge Computing (MEC) environments. The authors propose an Incentive Score Function (ISF) that integrates training contribution, data diversity, and communication latency into a reward function used to rank and reward clients. The authors of [79] propose a fairness-aware incentive mechanism for multi-server FL in edge-enabled wireless networks, with built-in differential privacy guarantees. Clients are evaluated via a multi-weight scoring algorithm that incorporates training accuracy, cost, and privacy noise level, and rewards are adjusted to ensure balanced distribution across heterogeneous participants.

The work in [69] proposes an incentive mechanism for FL using a Stackelberg games in industrial settings. The authors model the federated server as the leader and the clients as the followers, and the equilibrium of the game provides the optimal reward strategy for the server and the optimal contribution levels of the clients.

Furthermore, auction-based strategies have been explored to facilitate client selection and resource allocation in FL. The authors of [80] propose a utility-maximizing bidding strategy for data consumers in auction-based federated learning. Their approach distinguishes between the bid price and the actual cost paid under the generalized second-price (GSP) auction. To determine optimal bids, the authors employ a multitask learning framework that jointly estimates utility, market price, and winning probability, enabling ROI-based bid computation in a forward auction setting. In [81], the authors propose a budget-feasible double auction mechanism for FL markets, particularly in cross-silo FL settings. Their mechanism enables users to act as both providers and requesters, dynamically allocating roles based on needs and resources.

In [82], the authors consider model exchange between clients in cross-silo FL as economic trading and introduce a market design based on auction principles. The approach looks at each client's local model as an asset in a marketplace that can get monetary rewards if it gets used as part of the global model's aggregation process by the federated server, which plays the broker's rule. The work in [83] addresses the challenges faced in FL caused by variations in local model quality because of differences in data distribution between clients. To mitigate this, the authors propose an auction-based approach which dynamically selects resource-efficient clients and local models, minimizing resource usage while meeting quality requirements. The authors of [84] investigate FL in the wireless communication scenario and propose an auction mechanisms to facilitate trading between the federated server and clients. They define a data quality function using the earth mover's distance (EMD) to measure data distribution impacts for clients and propose an auction framework to optimize the social welfare of the FL environment.

Reputation systems are essential for maintaining trust and ensuring the reliability of participants in FL. The authors of [85] propose a decentralized FL system that combines clients' reputation with techniques like secure aggregation using homomorphic encryption and verifiable secret sharing, and in [86], the authors introduces the Federated Reputation Evaluation Blockchain, a framework that evaluates participants' reputations based on four factors: model contribution, data quality, participation frequency, and historical behavior. In [87], the authors introduces a credibility management scheme that assigns dynamic reputation scores to clients based on their historical contributions and behavior. The system incorporates time decay and attitudinal value factors to adjust reputation weights, enhancing the robustness of FL against adversarial attacks while maintaining low computational complexity.

Meanwhile, accurately assessing the value of client data is critical for fair incentive distribution in FL. In [88], the authors provide a comprehensive overview of existing methods for evaluating data contributions in FL. The work highlights various approaches, including Shapley value-based techniques and influence functions, highlighting their advantages and limitations. The paper also identifies open challenges in the field and suggests directions for future research to improve data valuation methods in FL. In [89], the authors present a framework that combines FL and Blockchain by utilizing the Shapley value to achieve a balance between data value quantification and privacy preservation. The approach aims to provide a practical method for assessing the value of data assets in FL environments, facilitating fair compensation and incentivization for data contributors. The authors of [90] introduce a privacy-preserving method for evaluating client contributions without a pre-specified training algorithm, enhancing transparency in data valuation. Their approach utilizes Wasserstein distance within the federated context, offering a practical solution for data valuation in FL. Additionally, the authors of [91] develop DDVal, a decentralized data valuation method capable of valuing individual data points in FL, offering scalability and applicability in non-IID data scenarios. Their method is based on sharing deep features and approximating Shapley values through a k-nearest neighbor approximation method. Finally, in [92], the authors introduce FedAVE, an adaptive data value evaluation framework designed to enhance collaborative fairness in FL. It dynamically assesses the value of data contributed by each participant, ensuring equitable model performance across diverse clients.

Concluding Remarks

The reviewed literature highlights the progress in addressing challenges within federated learning, including non-IID data distributions, selfish client behavior, and the application of game-theoretic models for client selection. Nevertheless, prior efforts are compartmentalized, treating these issues as independent research tracks rather than interdependent dimensions of a unified ecosystem. Such fragmentation limits the capacity of current frameworks to capture the coupled dynamics between trust, incentives, data heterogeneity, and system participation. This thesis departs from that view by advancing a holistic, multi-layered perspective that conceptualizes FL as a decentralized socio-technical economy composed of strategic, heterogeneous, and autonomous participants. Through the integration of trust management, coalition formation, incentive compatibility, and realistic benchmarking into cohesive theoretical and experimental frameworks, this thesis establishes a unified foundation for the sustainable, equitable, and practical deployment of FL systems.

From a data heterogeneity perspective, prior studies have explored various mitigation strategies, such as data augmentation, and personalized model components. While these approaches yield improvements in convergence and accuracy under non-IID settings, they often assume static data structures and overlook the interplay between data utility and system-level decisions. Our work contributes a more dynamic treatment of non-IIDness by embedding data utility into client selection and incentive structures, ensuring that both statistical and system heterogeneity are jointly considered.

In the context of selfish behavior, literature often treats free-riding as a threat to fairness and model integrity, proposing mechanisms such as exclusion thresholds, auditing, or reward-based filtering. While effective in principle, these methods often reinforce a centralized control narrative, where the server is the sole enforcer of fairness. Our approach rethinks this paradigm by abstracting the client selection process from the server and embedding it into a multi-agent game among clients themselves. This not only decentralizes selection, improving scalability and robustness, but also allows for a more nuanced, trust-based, and utility-aware coalition formation strategies.

On the other hand, the application of Stackelberg games in FL has generally modeled the server as the leader and clients as passive followers who optimize local training decisions based on server incentives. However, this top-down formulation has inherent limitations in fostering decentralized collaboration, especially in environments with asymmetric resources or limited server visibility. Our work contributes a novel Stackelberg formulation that operates among clients, where resourceful clients assume leadership roles and less-capable clients follow by joining their coalitions. This fosters an incentive structure that naturally supports inclusivity, data diversity, and trust propagation while maintaining theoretical rigor through equilibrium guarantees.

In terms of incentives and auctions, recent research has focused on designing payment mechanisms that ensure truthful bidding, resource-efficient participation, and fairness. Yet these systems either rely on heavy cryptographic primitives, assume trusted coordinators, or neglect the integration of reputation systems. Our proposed reverse auction-based FL marketplace introduces a cohesive and modular framework in which task owners, network operators, and clients interact through transparent bidding, coalition formation, and reward distribution mechanisms. Reputation is embedded not as a side feature but as a core utility component, influencing both leader formation and client selection dynamics. Furthermore, by clearly separating the roles of the task owner, network operator, and clients, our design avoids conflict of interest and reflects real-world economic and operational constraints.

CHAPTER 3 ARTICLE 1: COALITIONAL FEDERATED LEARNING: IMPROVING COMMUNICATION AND TRAINING ON NON-IID DATA WITH SELFISH CLIENTS

Accepted for publication in IEEE Transactions on Services Computing on February 15, 2023. Authors: Sarhad Arisdakessian, Omar Abdul Wahab, Azzam Mourad, Hadi Otrok.

3.1 Introduction

Despite the advantages of federated learning (FL) in preserving privacy and reducing raw data transfer (Chapter 1), practical deployments remain challenged by data non-IIDness, communication bottlenecks, and behavioral heterogeneity. Among these issues, selfish (or passively malicious) clients[1] minimize their local effort while still benefiting from the aggregated global model. Existing remedies primarily modify the learning pipeline while keeping decision-making server-centric [7]. However, they do not explicitly regulate how clients evaluate one another or how selfish behavior should influence participation.

To tackle these challenges, in this chapter we propose a novel Coalitional Federated Learning (CFL) paradigm, with a client-to-client trust component whose primary goal is to identify and isolate selfish clients. In our approach, trust is established from objective resource-use signals (under-utilization as a proxy for free-riding), and subjective peer recommendations, and then aggregated into trust scores. These scores inform a hedonic coalition formation process in which devices autonomously choose coalitions they prefer, and optimizes their utility. Within each coalition, a coalition master (CM) is elected to aggregate locally and serve as the single interface to the server, reducing communication bottlenecks.

It is worth noting that, the central server [2] still has authority, as it orchestrates phases, monitors resources, publishes objective trust signals, enforces participation policies, and selects which coalitions train and contribute to global aggregation. However, on top of this, our approach adds a client-to-client trust as a new metric alongside server-side signals, thereby limiting server bias that can arise from a single centralized viewpoint. This allows in democratizing participation as clients provide peer-sourced evidence and reveal preferences over coalitions, giving honest but previously under-recognized devices autonomy in admission and grouping.

Operationally, the server compiles objective trust from monitored resource histories and pub-

[1]Throughout, we use “selfish clients” or “free-riders” to denote passive malicious participants that conserve resources (CPU, RAM, bandwidth) by under-training or sub-sampling, without actively attempting to poison the model.

[2]Throughout, we use the terms central server and federated server interchangeably.

lishes them; in their turn, clients gather a bounded set of peer recommendations and combine both sources into aggregated trust scores during coalition formation. Devices then decide to join or leave coalitions based on their trust-based preferences; once coalitions stabilize, each coalition elects a CM. Training proceeds at the coalition level: members train locally, the CM aggregates and communicates the coalition update to the server, and the server aggregates across CMs. This design filters out selfish clients, and remains compatible with standard optimizers and existing techniques for non-IID data and communication [7].

3.2 Selfish Clients

In this section, we explain the notion of selfish devices and their behavior, and discuss the consequences of this behavior on the FL process. We also summarize the annotations and symbols used in this chapter in Table 3.1.

FL heavily relies on clients' resources to derive robust machine learning models. This implies that systems with abundant high quality data, will have better federated global models. The default approach [1] assumes that the devices selected to participate in the learning rounds are honest and willing to train on their resources (or at least with the capacity that they have advertised). In an FL scenario, an entire dataset is denoted by $D^Q = \{D_1^q ... D_n^q\}$ where D_n^q refers to the pieces of information a client owns pertaining to a specific work or a task $Q = \{1, 2, 3..., n\}$. Therefore, the learning process is heavily reliant on the resources of the underlying devices. Each device does its best to train the local model with the data it possesses by dedicating resources to it such as CPU, memory, and bandwidth. In return, these devices receive an enhanced version of the trained model, which is an aggregation of several local models into global one. Thus, devices benefit from each other's models in a cooperative way since each device's local model is expected to enrich the global model.

However, the FL process is likely to be challenged by the existence of selfish devices, which are not willing to perform the training with their full resources. Such selfish devices would still want to reap the benefits of the learning done by others through constantly receiving the updated versions of the global model. Selfish devices are different from *straggler devices* in FL [7], which take longer than usual to submit their model updates to the federated servers. Such devices cause significant delays in the model aggregation. The main difference between straggler devices and selfish devices is that the behavior of the former devices stems from some lack in computational and/or network resources, while the behavior of the latter devices is to obtain the FL model without wasting their resources. Hereafter, throughout this work, we mainly focus on selfish/free-rider devices.

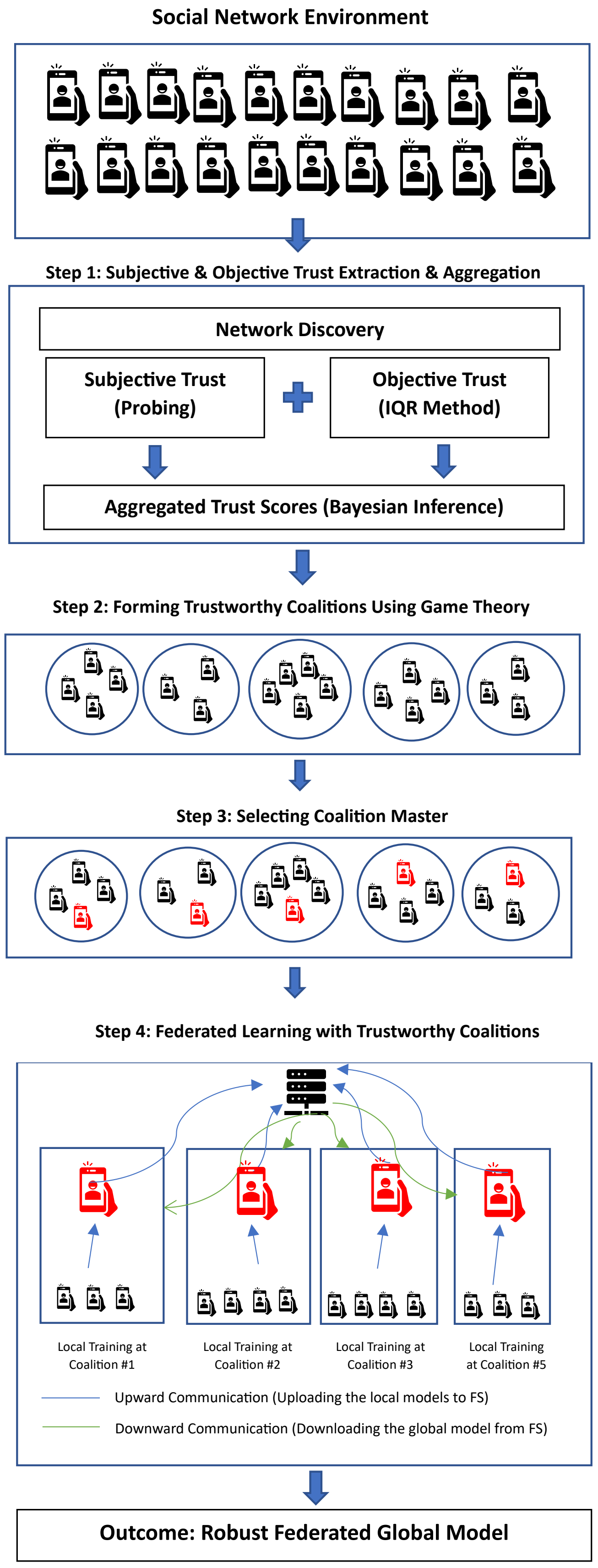

Social Network Environment
Step 1: Subjective & Objective Trust Extraction & Aggregation
Network Discovery
Subjective Trust (Probing)
Objective Trust (IQR Method)
Aggregated Trust Scores (Bayesian Inference)
Step 2: Forming Trustworthy Coalitions Using Game Theory
Step 3: Selecting Coalition Master
Step 4: Federated Learning with Trustworthy Coalitions
Local Training at Coalition #1
Local Training at Coalition #2
Local Training at Coalition #3
Local Training at Coalition #5
Upward Communication (Uploading the local models to FS)
Downward Communication (Downloading the global model from FS)
Outcome: Robust Federated Global Model

Figure 3.1 A high level representation of the proposed framework

Abbreviations	
D	Dataset of a client
Q	Set of all works/tasks
q	A work/task from Q
W	Set of all weights
w	A weight from W
T	Time Slots
t	A specific time slot from T
I	Set of devices
i	A device from I
Z	A coalitions structure
C	A coalition in Z
E	Set of edges between nodes
Rec	Recommendation of trust towards a device i
CM	Coalition Master
FS	Federated Server (used interchangeably with CS)
CS	Central Server (used interchangeably with FS)
P_i	Preference function of i
R	Set of all resource metrics {cpu, ram, bandwidth}
BW	bandwidth of a device
LC	Communication latency
Q1	1st quartile or 25th percentile
Q3	3rd quartile or 75th percentile
IQR	Interquartile Range
p	A value of current resource utilization of i
LL	Lower limit resource threshold
α	Total of under-utilized resources
ρ	Number of times a resource was under-utilized
ϕ	Average of under-utilizing a resource
η	Probability of under-utilizing a resource
δ	Count of type of resource under-utilized
τ_i	Objective trust value towards device i
$h_i(t)$	Historical set of coalitions for i before t

Table 3.1 List of symbols and definitions used in Chapter 3

The presence of such devices can entail negative consequences on the whole FL process:

- First and foremost, it would be unfair for the honest devices in the environment that are participating and dedicating their resources to receive the same global model as the selfish devices. This might demotivate honest devices from participating in further rounds and might even push them to entirely leave the network.
- The convergence of the global model will be delayed. It would take longer for the global model to converge to a certain desired accuracy. This will require the honest devices to spend more resources over a additional number of communication rounds to attain the desired accuracy.

3.3 Trust Model

In FL, the central server handles the client selection process. This would mean that the server relies on its own trust knowledge of the clients for making the selections. However, this knowledge can be limited, or biased towards certain clients. In the context of our framework, we define trust as a quantifiable measure that a client will behave honestly and cooperatively during the FL process. Specifically, a trustworthy client is one that dedicates its resources to local training, rather than strategically under-training for selfish resource savings, and has demonstrated positive cooperative behavior in this regard in prior coalition formations as acknowledged through peer feedback. However, most works in the literature mainly focus on the server-client trust aspect. This is where the notion of client-to-client trust is highlighted, as these relationships are key enablers for any distributed communication architecture in FL. This is because in addition to server's feedback regarding a client, we take the opinion of other clients in the environment that have dealt with the node in question and aggregate both trusts inputs accordingly. This decreases the possible bias towards classification of clients as honest or not as now more entities are involved in the classification process. Each client can have a separate and independent trust perspective regarding the other device. For example, a device 'A' can be highly trusted by a device 'B' because of a good past collaboration, yet not so much by another device 'C'. Hereafter, in this section, we explain the details of our trust model which allows the clients to autonomously establish trust relationships toward one another.

Our framework assumes that devices are able to exchange recommendations with peers, which may not always be possible due to intermittent connectivity, bandwidth constraints, or asynchronous participation. To address this, we design the trust aggregation mechanism such that objective trust can serve as a reliable prior even in the absence of peer feedback. Subjective recommendations are incorporated when available. This ensures practicality and scalability in real-world FL environments. Furthermore, new devices receive a neutral objective score of 1, ensuring they are not penalized upon entry. In early rounds, their subjective trust remains undefined, and hence their influence on coalition formation is minimal. As they participate in training and their telemetry becomes available, their objective and subjective trust are recalibrated. This approach balances cold-start fairness where devices are not excluded, as well as progressive accountability where trust becomes data-driven over time.

3.3.1 System Model

We consider a set of devices $I = \{i_1, \ldots, i_n\}$. Each device $i \in I$ is characterized by a resource vector over CPU, RAM, and network bandwidth, $R = \{\mathrm{CPU}, \mathrm{RAM}, \mathrm{BW}\}$, with time-varying observations collected during training rounds.

Inter-device relations are represented by a logical, directed overlay $G_t = (I, E_t, \mathrm{Trust}_t)$, which may be partially known or initially absent and can be refined as the system operates. An edge $(i_k, i_j) \in E_t$ indicates that device i_k may draw on device i_j's past experience (e.g., prior co-participation or exchanged recommendations) when forming an opinion about i_j. The associated trust value $\mathrm{Trust}_t(i_k, i_j)$ is generally asymmetric (i.e., $\mathrm{Trust}_t(i_k, i_j) \neq \mathrm{Trust}_t(i_j, i_k)$) and may be missing at initialization. Trust values evolve through: (1) objective signals derived from locally measurable resource usage during training, and, (2) subjective endorsements obtained from other devices that have interacted with the target.

3.3.2 Objective Trust: Resource Monitoring

Our objective trust approach is based on monitoring the resource utilization of the client devices for certain FL tasks and comparing this utilization with historical utilization on similar tasks. Technically speaking, our approach capitalizes on the Interquartile Range (IQR) technique to check for any abnormal utilization of resources using two disjoint quartiles referred to as $Q1$ and $Q3$. Our motivation for choosing IQR is due to its lightweight nature as well as its minimal time and space complexity [93]. The aim here is to find those devices that are under-utilizing their resources throughout the FL process, indicating that they might be acting in a selfish manner by not dedicating enough resources to this process.

The algorithm takes as input the historical utilization traces of the FL-relevant resources (CPU, RAM, bandwidth) for device i over time slots T, and outputs the objective trust score τ_i (Lines 1–2). It initializes two accumulators: `sum_eta`, which will sum per resource under-utilization severities, and δ, the number of resource types that exhibited under- utilization

Algorithm 1 Objective Trust Calculation Algorithm

```
1: Input: Historical values of resources $R = \{\text{CPU}, \text{RAM}, \text{BW}\}$ for device $i$ over $T = \{t_1, \ldots, t_k\}$
2: Output: $\tau_i$ : objective trust value toward the device $i$
3:
4: Method: CALCULATE-OBJECTIVE-TRUST
5:     Initialize sum_eta $\leftarrow 0$;  $\delta \leftarrow 0$
6:     for each resource $r \in R$ do
7:         Compute $Q1_r$, $Q3_r$, $IQR_r$
8:         Compute $LL_r \leftarrow Q1_r - 1.5 \times IQR_r$
9:         Initialize $\alpha \leftarrow 0$;  $\rho \leftarrow 0$
10:        for each usage value $p$ at $t \in T$ do
11:            if $p < LL_r$ then
12:                $\alpha \leftarrow \alpha + p$
13:                $\rho \leftarrow \rho + 1$
14:            end if
15:        end for
16:        if $\rho > 0$ then
17:            $\phi_r \leftarrow \alpha/\rho$
18:            $\eta_r \leftarrow \phi_r / \max(LL_r, \varepsilon)$
19:            sum_eta $\leftarrow$ sum_eta $+ \eta_r$
20:            $\delta \leftarrow \delta + 1$
21:        end if
22:    end for
23:    if $\delta = 0$ then
24:        $\tau_i \leftarrow 1$
25:    else
26:        $\tau_i \leftarrow$ sum_eta$/\delta$
27:    end if
28: End method
```

at least once (Line 5). Then, for each resource $r \in \{\text{CPU}, \text{RAM}, \text{BW}\}$, it computes $Q1_r$, $Q3_r$, the interquartile range IQR_r, and the lower utilization limit $LL_r = Q1_r - 1.5 \cdot IQR_r$ following Tukey's rule (Lines 6–8). Next, it scans the history of r to collect all *under-utilization events* $p < LL_r$, accumulating their sum α and count ρ (Lines 9–13). After the scan, if $\rho > 0$ the algorithm summarizes the severity for this resource by computing the mean of the flagged samples $\phi_r = \alpha/\rho$ (Line 17) and converting it to a dimensionless severity ratio $\eta_r = \phi_r / \max(LL_r, \varepsilon) \in (0, 1]$ with a tiny $\varepsilon > 0$ for numerical safety (Line 18). The algorithm adds η_r to `sum_eta` and increments δ (Lines 19–20). Finally, if no resource was ever under-utilized ($\delta = 0$), it sets $\tau_i = 1$ (Lines 23–24); otherwise it returns the per resource average $\tau_i =$ `sum_eta`$/\delta$ (Lines 25–26). This construction ensures $\tau_i \in (0, 1]$: values near 1 indicate normal utilization, while values near 0 indicate frequent and/or severe under-utilization.

3.3.3 Subjective Trust: Recommendation Collection

To mitigate the central server's dominance in client selection and allow clients to actively participate in the evaluation process, we introduce a client-to-client trust mechanism for gathering behavioral recommendations among devices. This is done by consulting the devices that have a certain connection and have formed a certain sense of trust (or the lack of) in regards to their neighbors with which they have interacted in the past. The advantage of such an approach is its ability to take into account the recommendations of several devices rather than a single entity that can hold a false or biased recommendation about some other device. This type of trust calculation is known as *subjective* trust. To accomplish this, we devise an algorithm (Algorithm 2) that allows devices to inquire about others. This is done by consulting devices that have dealt with the device in question in the past and have a certain knowledge about its behavior.

Algorithm 2 Subjective Feedback

```
1: Input: origin device $orig$, target device $tgt$
2: Output: list $R$ of recommendation scores about $tgt$

   $\textsc{GetNeighbors}(orig, tgt) \to N$
   $\textsc{RequestRec}(n, tgt) \to rec \in [0, 1]$
3: $R \leftarrow [\ ]$
4: $N \leftarrow \textsc{GetNeighbors}(orig, tgt)$
5: for each $n \in N$ do
6:     $s \leftarrow \textsc{RequestRec}(n, tgt)$
7:     Append $(n, rec)$ to $R$
8: end for
9: return $R$
```

Algorithm 2 takes as input the origin device initiating the request and the target device being evaluated, and outputs a list R of neighbor-provided recommendation scores about the target (Lines 1–2). The procedure is agnostic of the type of the environment and exposes two abstract interfaces: $\textsc{GetNeighbors}(orig, tgt) \to N$, which returns a set of neighbors that have interacted with the target, and $\textsc{RequestRec}(n, tgt) \to rec$, which returns a trust recommendation score from neighbor n. The algorithm initializes an empty list R (Line 3), obtains the neighbor set N via $\textsc{GetNeighbors}$ (Line 4), and then iterates over each $n \in N$, requesting a recommendation score rec and appending the pair (n, red) to R (Lines 5–7). Finally, it returns the accumulated list of neighbor feedback R (Line 9).

3.3.4 Trust Aggregation

After having derived the subjective and objective trust values for the devices, we propose an aggregation formula to aggregate both trust sources. The formula is inspired by the aggregation models discussed in [94, 95]. We adopt a Bayesian-inspired aggregation, treating objective trust as a prior belief and peer recommendations as evidence. Equation 3.1 corresponds to the posterior mean with equal weight given to prior and observed recommendations, motivated by the need to decrease server's bias on the final results, and provide fairer recommendations by letting clients have their say in the final answer. This ensures that the server's view of a client (through objective trust) and the collective opinions of peers (through subjective trust) contribute with equal weights. Conceptually, this structure reduces the central server's grip on client selection, since it can no longer impose its own assessment unilaterally. Instead, neighbor devices collectively influence the trust estimation process, thereby democratizing the system. In this formulation, the objective trust value τ_i is treated as the prior, with its strength parameter implicitly set equal to the number of peer recommendations n. This choice ensures that the objective trust contributes of equal weight to the subjective recommendations, allowing Equation 3.1 to be interpreted as the posterior mean of a Bayesian aggregation.

$$Trust_i = \frac{n\tau_i + \sum_{j=1}^{n} Rec_j}{2n} \tag{3.1}$$

Where $Trust_i$ represents the final trust score computed for the device i, τ_i is the objective trust value toward device i obtained from Algorithm 1, Rec_j denotes each individual recommendation about device i received from the peers during the subjective trust phase using Algorithm 2, and n is the total number of such recommendations collected.

3.4 Coalitional Federated Learning

In this Section, we first formulate the coalitional game, including the preference function, that enables the client devices to form efficient coalitions and then we explain the practical implementation of our coalitional federated learning paradigm.

3.4.1 Utility and Preference Functions

A Hedonic coalitional game is a type of cooperative game that analyzes the interactions between the players where they seek to form coalitions and be part of them. In such games, the players have the privilege to choose which coalition to join as they have full control over their preference rankings. In such games, the players are rational and self-interested, accordingly they are concerned about the identity of other players in their coalition as it might affect their utility.

Based on the above, an important part of the coalitional game is the utility on which the players rely to adjust their preferences. Let $U_i(C)$ denote the utility of the device i vis-à-vis a given coalition C. The utility of i is calculated by summing up i's beliefs in the trustworthiness of each member j of C using Equation 3.2.

$$U_i(C) = \frac{1}{|C|} \sum_{j \in C} \text{Trust}(j). \tag{3.2}$$

Equation 3.2 defines the utility of a coalition C as the average trust of its members. Here, C denotes the set of participating devices, $|C|$ represents its cardinality (i.e., the number of members).

Intuitively, client devices prefer to join coalitions that exhibit a higher level of trust, thus minimizing the number of selfish members. The objective is to be part of strong and honest coalitions, where each of the devices effectively contributes to the training of the FL model. In this context, the devices that join coalitions consisting of a large number of selfish members would end up having lower trust scores. Accordingly, honest devices in a coalition with a large number of selfish members might even need to spend more resources and participate in more communication rounds to compensate the harm caused by their selfish counterparts.

For every device $i \in I$, a preference relation denoted by $(\succeq_{i \in I})$ is a complete, reflexive, and transitive binary relation over the set of all possible coalitions that i could join. Using this relation, $C \succ_i C'$ implies that device i strictly prefers to be member of coalition C over C', while $C \succeq_i C'$ implies that i prefers to be member of coalition C over C' or has no preference between the two coalitions. Based on this definition, the preference relation of each device is given as follows:

$$C \succeq_i C' \iff P_i(C) \geq P_i(C') \tag{3.3}$$

where C and C' being any two coalitions of which device i could be a member, and P_i being the preference function for any device i such that:

$$P_i(C) = \begin{cases} -\infty, & \text{if } i \text{ believes that } C \text{ contains selfish members} \\ 0, & \text{if } C \in h_i(t) \\ U_i(C) & \text{otherwise} \end{cases} \tag{3.4}$$

where $h_i(t)$ represents the history set of device i at time t, containing the coalitions that i has joined previously and left at a time $t^{'} < t$ prior to the establishment of the present coalition structure [96]. The preference function defined in Equation 3.4 enables each device i to join the coalition that maximizes its trustworthiness. At the same time, it enables the devices to avoid the coalitions that they believe contain selfish devices, based on our proposed client-to-client trust framework. This is why the function assigns a $-\infty$ preference to any coalition that the device believes contains selfish devices. Furthermore, each device tries as much as possible to avoid joining any previously joined coalition in which there have not been any changes in terms of its members. This is possible by assigning the value 0 to any coalition that is part of the device's history set. Lastly, the device would prefer to join a coalition that maximizes its utility, which quantifies the overall device's trust toward the coalition's members.

3.4.2 Federated Learning Coalition Formation

To derive the equilibrium of the game, we first formally define the coalition structure and propose a distributed coalition formation algorithm for the devices (Algorithm 3). This algorithm enables the devices to decide about the coalitions to join so as to maximize the overall trust.

Convention. Let Z be a partition of the device set I into finite coalitions $\{C_1, \ldots, C_m\}$. For each device $i \in I$, let $C(i)$ denote the unique coalition in Z that contains i. Player i's preference $\succeq_i$ is defined only over coalitions that contain i; when C does not contain i we compare $C(i)$ with $C \cup \{i\}$. We also allow $C = \varnothing$, where $C \cup \{i\} = \{i\}$ denotes forming a singleton.

Definition 1: A coalition structure is a set of coalitions $Z = \{C_1, ..., C_j\}$ that divides the set I of devices into disjoint coalitions in a way that $\forall i \neq i', C_i \cup C'_{i'} = \emptyset$. We denote by $C(i,t)$ the coalition containing client i at time t (shorthand $C_i(t)$).

We explain in Algorithm 3 the distributed client devices coalition formation process. The

Algorithm 3 Coalition Formation

```
1: Input: Initial partition $Z(t) = \{C_1(t), \ldots, C_m(t)\}$ at time $t$
2: Output: Final coalition structure $Z^*(t_f)$ at time $t_f$
3:
4: Method: COALITION-FORMATION
5: initialize $t \leftarrow 0$
6: initialize $Z(t) \leftarrow \{C_1(t), \ldots, C_m(t)\}$
7: initialize $h_i(t) \leftarrow \emptyset$   for all devices $i$
8: repeat
9:     for each device $i \in \bigcup Z(t)$ do
10:        let $C(i,t)$ be the current coalition containing $i$
11:        select a coalition $C_\ell(t)$ such that $i \notin C_\ell(t)$
12:        compute $P_i(C(i,t))$ using Eq. 3.4
13:        compute $P_i(C_\ell(t) \cup \{i\})$ using Eq. 3.4
14:        if $P_i(C_\ell(t) \cup \{i\}) > P_i(C(i,t))$ then
15:            leave $C(i,t)$ and join $C_\ell(t)$
16:            update history $h_i(t)$
17:        elseif $P_i(C_\ell(t) \cup \{i\}) = P_i(C(i,t))$ then
18:            stay in $C_i(t)$
19:        else
20:            $Z(t{+}1) \leftarrow Z(t)$
21:        end if
22:    end for
23: $t \leftarrow t + 1$
24: until no change occurs in the partition
25: $Z^*(t_f) \leftarrow Z(t)$
26: return $Z^*(t_f)$
27: End method
```

algorithm takes as an input the initial partition of the coalition structure (Line 1) and outputs the final coalitional structure (Line 2). Lines 4−7 are used for initialize the different variables. Thereafter, the algorithm loops over each device in the coalition structure (Line 9), saving the coalition of which they currently are member (Line 10). It then arbitrarily selects a coalition of which the underlying device is not member (Line 11). Then, based on equation 3.4, the algorithm calculates the preference P_i of the coalition that contains i (Line 12) and that of the other arbitrarily chosen coalition that i can potentially join (Line 13). Based on the computed trust values, the algorithm compares the utility values of both coalitions (Line 14). In case the new coalition has a higher utility compared to the current one (Line 15), the device would prefer to leave its current coalition and join the new one, while updating its history set (Line 16). If the preferences between the two coalitions are the same, the device i stays in its current coalitions (Line 18), Else, the coalition structure remains the same at the current time moment, and that coalition structure is assigned to the next time moment (Line 20). The algorithm is repeated until no change in the partitioning of the coalitions occurs

(Line 24), meaning that the algorithm has converged to a Nash and individual stable points. Note that, according to the proposed algorithm, the selfish devices will end up in singleton coalitions (due to the preference function proposed in Equation (3.4)). This greatly helps the federated server avoid the selfish devices when making its selections.

Definition 2 (Nash stability). A coalition structure Z is *Nash stable* if no device can profitably deviate unilaterally to any existing coalition or to a singleton. Formally,

$$\forall\, i \in I,\ \forall\, C \in Z \cup \{\varnothing\} : \quad C(i) \ \succeq_i \ C \cup \{i\}.$$

Definition 3 (Individual stability). A coalition structure Z is *individually stable* if there is no device i and coalition $C \in Z \cup \{\varnothing\}$ such that

$$C \cup \{i\} \ \succ_i \ C(i) \quad \text{and} \quad \forall\, j \in C : \ C \cup \{i\} \ \succeq_j \ C.$$

Based on these two definitions along with the proofs provided in [97], Algorithm 3 converges to a final coalition structure $Z^*(t_f)$ which consists of several disjoint coalitions that are both Nash and individually stable.

3.4.3 Coalition Master Election

Once the coalitions are formed, the devices belonging to the same coalition share resource information with one another. We treat the transport of pre-selection summaries as an implementation detail; any topology-appropriate secure channel can be used, provided messages are authenticated and delivered within the allocated window. This information is used to elect a Coalition Master (CM) for each coalition. The master serves as the point of contact between its coalition members and the federated server. It is mainly responsible for (1) aggregating the local models trained by its coalition members; (2) sending the aggregate coalition-level models that are locally trained in its coalition to the federated server; and (3) downloading the latest version of the global model from the federated server and sharing it with its coalition members. This is important to avoid pairwise communications between each client and the federated server, which are a cause for communication bottlenecks in FL.

The resource information shared by devices are focused on four resource metrics which are crucial for FL training, i.e, available CPU, RAM, bandwidth (BW) and latency (LC) between each device and the federated server. Formally, let CPU_i be the amount of CPU available at device i, RAM_i be the amount of RAM available at i, BW_i be the amount of bandwidth available at i and $LC_{i,CS}$ be the latency between device i and the central server CS. Also, let $W = \{w_{cpu}, w_{ram}, w_{bw}, w_{LC}\}$ be the set of weights representing the different weights for the metrics that are taken into account for the master's election. We also note that as an optional part, each coalition master can share some data pertaining to the task with the clients in the coalition to combat non-IIDness.

Before scoring, we normalize each metric per coalition to $[0, 1]$ and use $1-\widehat{LC}_{i,CS}$ for latency.

If multiple devices attain the same score, we break ties by (i) smaller normalized latency $\widehat{LC}_{i,CS}$, then (ii) larger normalized bandwidth $\widehat{\mathrm{BW}}_i$, then (iii) smallest device ID.

Let $Score(CMS_i)$ be the score obtained by each device i to quantify its adequacy in being the master of its coalition, and is calculated according to equation (3.5).

$$Score(CMS_i) = w_{\mathrm{cpu}}\widehat{\mathrm{CPU}}_i + w_{\mathrm{ram}}\widehat{\mathrm{RAM}}_i + w_{\mathrm{bw}}\widehat{\mathrm{BW}}_i + w_{\mathrm{lc}}\Big(1 - \widehat{\mathrm{LC}}_{i,CS}\Big), \tag{3.5}$$

with weights $w_{\mathrm{cpu}}, w_{\mathrm{ram}}, w_{\mathrm{bw}}, w_{\mathrm{lc}} \geq 0$.

The CM election algorithm is described in Algorithm 4.

Algorithm 4 Coalition Master Election

```
1:  **Input:** A coalition $C$
2:  **Input:** Set of devices in $C$
3:  **Input:** Set of weights for the resource metrics
4:  **Output:** Master of coalition $C$
5:
6:  **Method** COALITION-MASTER-ELECTION
7:      Devices in $C$ share their resource information
8:      **For each** device $i \in C$ **do**
9:          Compute $Score(CMS_i)$ using Eq. (3.5)
10:         Store $Score(CMS_i)$ in list $ListMaster$
11:     **End For**
12:     **Return** device with the highest score in $ListMaster$
13: **End method**
```

The algorithm takes as input a certain coalition, the set of devices inside that coalition, and the set of weights for the resource metrics (Lines $1-3$) and returns the elected coalition master (Line 4). First, the devices inside the coalition share with one another their resource information in terms of available CPU, RAM, bandwidth, and latency from the federated server (Line 7). This information is then used alongside the input weights to compute the score $Score(CMS_i)$ of each device using equation 3.5 (Line 9). The scores of all devices belonging to the same coalition are stored in a list $ListMaster$. Finally, the device with the highest score is returned and selected to be the coalition master (Lines 12).

We note that in case of a coalition master failure, the system can use the second highest ranking client from the output of algorithm 4 as the new coalition master, or can trigger a CM re-election within the coalition with the information it already has.

3.4.4 Coalition-based Federated Learning Training

Having formed the client devices' coalitions, we explain in Algorithm 5 how the practical FL training takes place using the proposed *Coalitional Federated Learning* architecture.

Algorithm 5 Trust-enabled coalitional federated learning

```
 1: Input: Set of devices, federated Server (FS)
 2: Output: Final model parameters of FL
 3:
 4: Method: TeC-FL:
 5:     Phase 1: Client-to-client trust establishment mechanism:
 6:         Run Algorithm 1 to derive the devices' objective trust
 7:         Run Algorithm 2 to derive the devices' subjective trust
 8:         Use Equation (3.1) to aggregate both sources of trust
 9:     Phase 2: Trust-Enabled Coalition Formation
10:         Run Algorithm 3 to form the client devices coalitions
11:     Phase 3: Coalition Master Selection
12:         Run Algorithm 4 to elect the CM of each coalition
13:     Phase 4: Coalitional Federated Learning
14:         For A given number of rounds, do
15:             CS selects non-singleton coalitions for training
16:             For each non-singleton coalition do
17:                 CM shares a subset of data with its members (optional)
18:                 Members merge local and shared data (optional)
19:                 Members start the training process
20:                 Members share model parameters with CM
21:                 CM aggregates local results
22:             End For
23:                 CM sends aggregate local models to FS
24:                 FS performs aggregation of local models
25:                 FS sends global model to the coalitions
26: End method
```

Algorithm 5 takes as input the set of the client devices and the federated server (Line 1). It outputs the final model parameters obtained at the end of the federated learning process (Line 2). The procedure is organized into four phases.

In Phase 1, the system establishes trust among clients (Line 5). Objective trust is derived from each device's resource usage history using Algorithm 1 (Line 6). Subjective trust is obtained from neighbors recommendations via Algorithm 2 (Line 7). These two sources are aggregated using Equation (3.1) to yield a final trust score for each device (Line 8).

In Phase 2, coalition formation is performed (Line 9). Algorithm 3 is executed, enabling devices to join coalitions according to their trust-based preferences, with selfish devices eventually isolated into singletons (Line 10).

In Phase 3, coalition masters (CMs) are elected (Line 11). Algorithm 4 is invoked inside each coalition to select the member with the highest weighted resource score as CM (Line 12).

In Phase 4, coalitional federated learning is carried out (Line 13). For a given number of training rounds, the federated server selects only non-singleton coalitions to participate (Lines 14-15). For each such coalition (Line 16), the CM may optionally share a subset of data to mitigate non-IIDness (Line 17), after which members merge local and shared data (Line 18) and perform local training (Line 19) [98]. Model updates are sent to the CM (Line 20), which aggregates them into a coalition-level model (Line 21). Once coalition aggregation is completed, the CM uploads the result to the federated server (Line 23). The federated server aggregates across all coalition updates to produce the global model (Line 24), which is then broadcast back to coalition masters for dissemination to their members (Line 25). This loop continues for the prescribed number of rounds, after which the algorithm outputs the final global model parameters as the outcome of the training process.

Complexity

Phase 1 (Trust establishment). In Phase 1, both objective and subjective trust values are computed. Objective trust is obtained at the server from historical CPU, RAM, and bandwidth, requiring $O(NRT)$ time to process T samples for R resources across N devices. Subjective trust is evaluated locally, with each device exchanging feedback with a number of neighbors. The total subjective-trust cost scales linearly with N. Aggregation of both components into a unified trust score adds $O(N)$ time, so the overall complexity of the trust-establishment phase remains $O(NRT)$ in practice.

Phase 2 (Coalition formation). Coalitions are then formed following the hedonic game procedure, where each device evaluates whether joining another coalition improves its utility and performs the switch only if it does. Although, in theory, a device could evaluate several alternative coalitions, the actual search space is small in practice since the number of active coalitions (C) in the environment is typically much smaller than the total number of participants ($C \ll K$). Moreover, the size of each coalition can be administratively bounded by the network owner to maintain balanced workloads and communication efficiency. As a result, each device only considers a few feasible destinations, and convergence is reached after a limited number of switches. Together with the history rule that prevents rejoining previously left coalitions, the overall cost of coalition formation remains near-linear in the number of devices, approximately $O(NC)$ in practice.

Phase 3 (Coalition master election). Once the coalition structure is fixed, each coalition C elects a master based on the resource capacities of its members. Devices in C exchange summaries of their resource information (e.g., CPU, RAM, bandwidth, and latency), and each member locally evaluates the same deterministic scoring function to rank all candidates. The device with the highest score is selected as the coalition master. Since the scoring and comparison operations scale linearly with the coalition size, and coalitions partition the N devices, the overall election cost across all coalitions is $O(N)$.

Once the coalition structure is finalized, standard federated training proceeds within and across coalitions, where local model updates are performed and aggregated according to the federated learning procedure.

3.5 Empirical Results and Discussion

In this section, we conduct a series of experiments pertaining to our work, present our results, and discuss the main findings.

3.5.1 Experimental setup

To perform the simulations, we build our own environment in which each device is equipped with a CPU, RAM, and network bandwidth capacity. We utilize Python 3 alongside several libraries such as Numpy, Torch, Torchvision, and Syft. We run the simulations on a workstation that is equipped with an Intel i7 processor with 16 GB of RAM. The operating system on which we base our simulations is Microsoft Windows 10. We use the MNIST (Modified National Institute of Standards and Technology database) dataset [99] which can be downloaded as part of the Torchvision datasets. MNIST contains 60,000 training and 10,000 testing images of $28x28$ pixel each. The images represent handwritten digits ranging between 0 and 9 in black and white alongside their labels. In the federated learning part, we use the *FedAvg* traditional local model aggregation method proposed in [1]. We change the number of devices between 20, 40, 60, 80, and 100. Furthermore, to classify a client as either selfish or not, we experimentally derive a threshold (Section 3.5.2), where any device having a trust of value (computed as per Equation 3.1) below this threshold is considered to be selfish. This methodology for deriving a trust threshold has been adopted in [100] where the authors experiment with different threshold values in the context of trust management using a Bayesian approach. We note that, for simplicity, our experimental setup assumes that devices do not straggle and instead focuses on detecting selfish behavior.

We compare our solution 'FedTrust' with the two following existing approaches:

1. Vanilla-FL: This is the de facto approach that assumes a fully honest client environment in the sense it does not take into account the presence of selfish clients that try to harm the FL process.

2. A State-of-the-art approach [51] that relies on trust between the federated server and the devices individually. This approach utilizes the resources of devices such as CPU and Memory resources coupled with DDQN (Deep Reinforcement Learning with Double Q-learning) technique to derive the trust scores between the two entities and. Hereafter, in our writings we refer to this approach as 'DDQN-FL'.

3.5.2 Threshold Value Determination

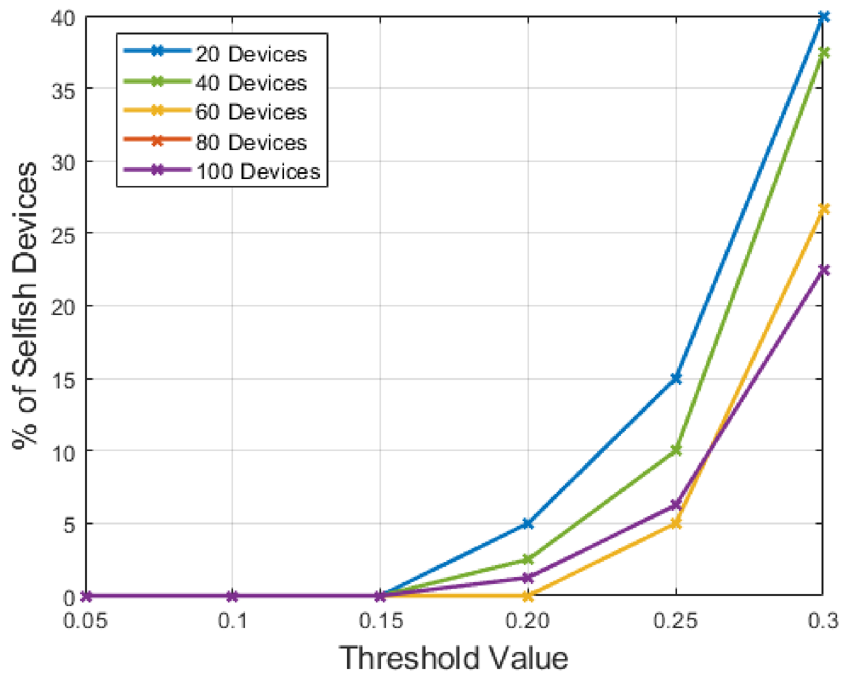


Figure 3.2 The effect of various threshold values on the selfish device detection in different environments

For the devices to evaluate each other's trustworthiness and select the coalitions that maximize their utility, a trust threshold should be derived that serves as an anchor for trust. That is, if a device's aggregate trust value (using Equation 3.1) is above that threshold, the device would be considered to be trustworthy; otherwise, the device would be considered to be selfish. Therefore, selecting an appropriate trust threshold is of utmost importance. Selecting an arbitrarily high threshold value would make the environment severe and would result in a large number of selfish devices in the network. On the other hand, selecting an arbitrarily low threshold value would result in an almost fully honest environment in which the number of selfish devices is minimal. To avoid these problems, we apply an experimental approach to derive a fair trust threshold. Specifically, we study the impact of several threshold values (i.e., 0.05, 0.10, 0.15, 0.20, 0.25, and 0.3) on the percentage of selfish devices in the environment while also varying the number of devices, as highlighted in Figure 3.2. We notice from the figure that with the values of 0.05, 0.1, and 0.15 for the threshold and for the different studied number of devices (i.e., 20, 40, 60, 80 and 100), the algorithm fails to detect selfish devices in the final coalitions. A threshold value of 0.2 can detect a small number selfish devices for an input number of devices of 40, 60, 80 and 100. On the contrary, a threshold value of 0.3 results in a large percentage of selfish devices. For example, for an environment with 20 devices, a threshold value of 0.3 results in 40% of the environment being selfish. To avoid these two extreme scenarios, we select the value of 0.25 a reasonable empirical choice. Thus, in the rest of the experiments, the value of 0.25 will be used for the trust threshold.

3.5.3 Percentage of Selfish devices in the Coalitional Federated Learning Environment

In this set of experiments, we compare the effectiveness of our approach against the other two baseline approaches in terms of minimizing the number of selfish devices in the coalitional FL environment. This, however, does not include the selfish devices that end up in singleton coalitions (i.e., the devices that do not get chosen by any other devices based on the preference function in Equation 3.4) as these singletons will be automatically avoided during selection by the federated server.

We notice in Figure 3.3 that our solution is successful in minimizing the number of selfish devices in final structure of the coalitional FL across the different input number of devices. On the contrary, the Vanilla-FL client selection approach entails the highest percentage of selfish client devices. It can be observed that our solution can reduce the percentage of selfish devices up to $4x$ to $5x$ more than Vanilla-FL approach. For example, in an environment consisting of 20 devices, our approach results in only 5% of selfish devices in the final coalition structure as opposed to 20% in the case of Vanilla-FL. Furthermore, as the number of devices increases, our solution continues to outperform the Vanilla-FL at a larger scale. Compared

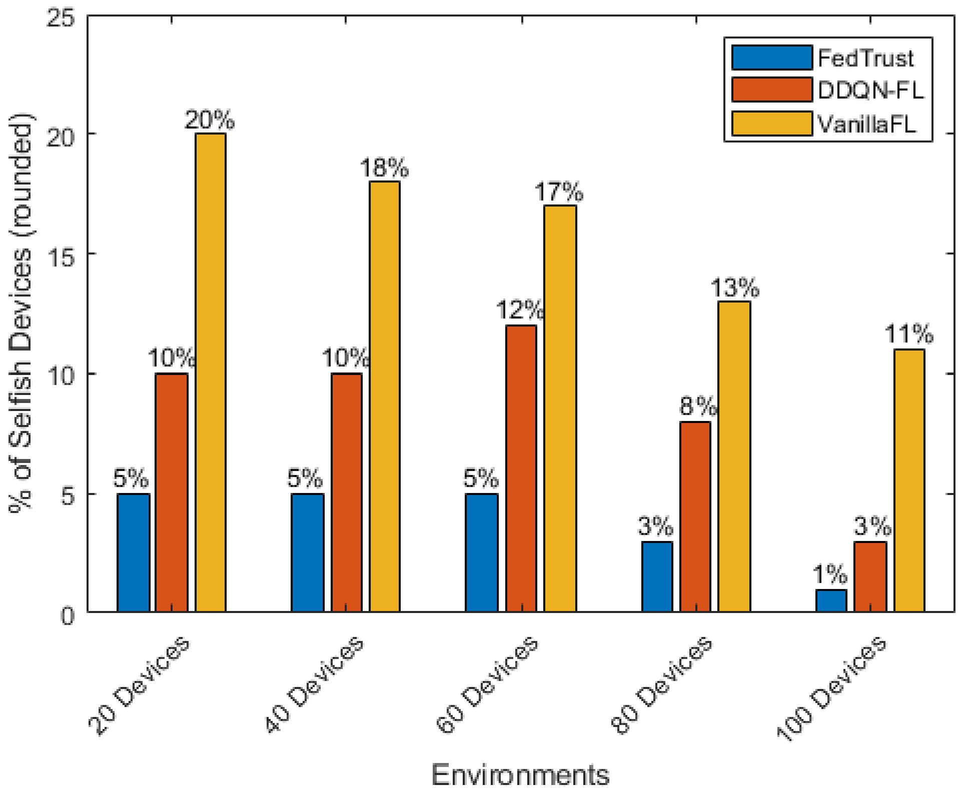
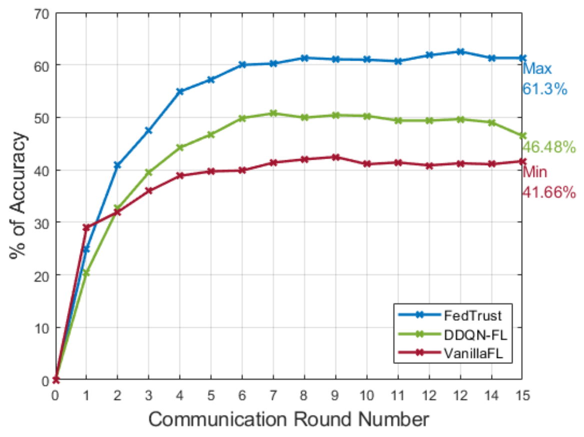


Figure 3.3 Percentage of selfish devices in the final coalitional structure

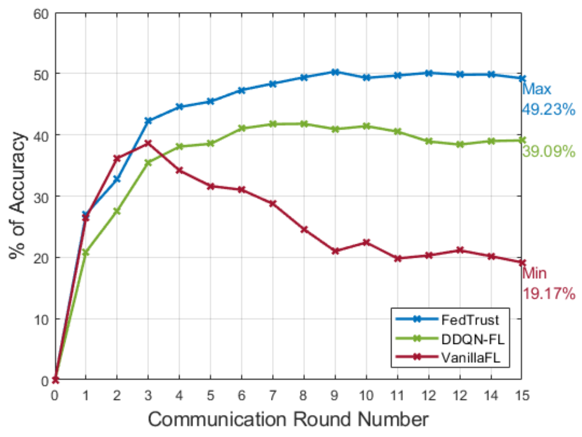
to the DDQN-FL approach as well, our approach reduces the percentage of selfish devices. For example in environments with 20, 40, and 60, we can see that our approach minimizes the percentage of selfish devices by $2x$ compared to the DDQN-FL approach and more than $2x$ in environments with 80 and 100 devices. The primary distinction lies in the fact that the DDQN-FL approach considers only the trust relationships between the federated server and the clients, without examining the trust that can emerge among the clients themselves. In contrast, our solution incorporates both inter-client and server–client trust, allowing for a more comprehensive and decentralized evaluation. Furthermore, unlike DDQN-FL, which derives trust solely from CPU and RAM capacities, our method also accounts for network bandwidth; a critical factor in federated learning, where communication efficiency between clients and the server plays a decisive role and should not be overlooked in trust computation.

3.5.4 FL Training Accuracy

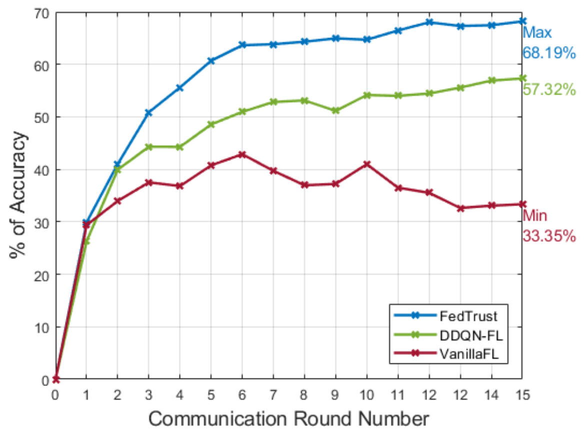
In Figure 3.4, we study the performance of the different comparison approaches in terms of improving the accuracy of the overall global FL model. In this set of experiments, we vary



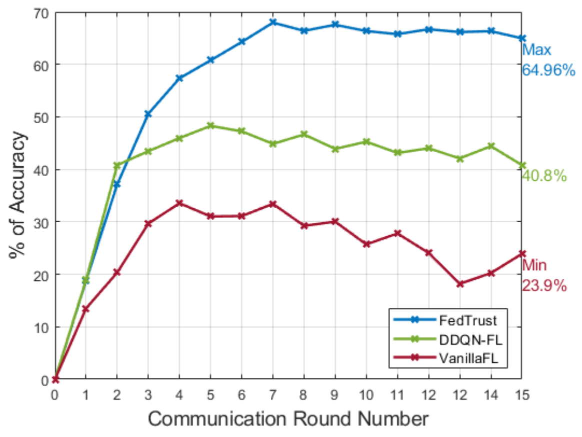
(a) 20 devices environment



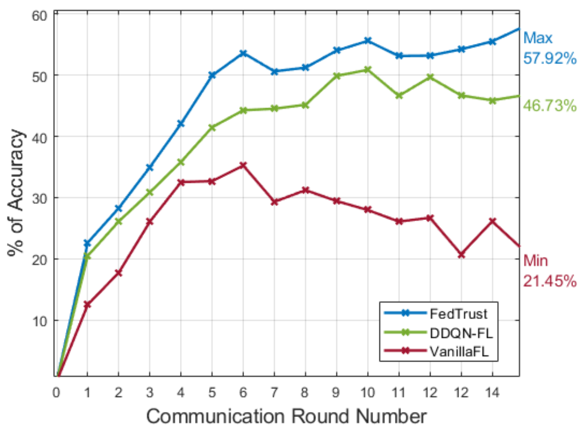
(b) 40 devices environment



(c) 60 devices environment



(d) 80 devices environment



(e) 100 devices environment

Figure 3.4 The effect of accuracy using FedTrust against baseline and state-of-the-art under various federated environments.

the number of client devices from 20 to 100.

We notice from figure 3.4 that our approach scores the best (maximum) in terms of accuracy compared to the other approaches. The results also indicate that the Vanilla-FL approach yields the least (minimum) accuracy, while the DDQN-FL method, although performs better than the traditional Vanilla-FL method, still falls short compared to our approach. Particularly, our approach attains maximum accuracy of 68% in the 80 devices environment which is around 24% higher compared to DDQN-FL, and around 41% higher than the vanilla approach. On the other hand, our approach scores the minimum in terms of accuracy in the 40 devices environment with an accuracy of 49% which is still around 10% higher than the DDQN-FL and 30% higher than the vanilla approach. The results highlight that in different and various environments, our approach keeps on scoring better results in terms of accuracy compared to the other approaches. An accuracy gap ranging between around 10% to 25% (Figures 3.4a, 3.4b, 3.4c, 3.4d, 3.4e respectively) between our approach and the DDQN-FL approach can be noticed. This gap stems from the fact that our approach takes both subjective and objective trust metrics into account and integrates an aggregation method for these trust metrics. Moreover, our solution enables the client devices to form trustworthy coalitions where the number of selfish devices is minimal using client-to-client trust relationships, while on the other hand, the DDQN-FL approach does not take into account the cross-client trust relationships. Furthermore, the difference in accuracy between our approach and the Vanilla-FL approach varies between around 20% and 40% and more (Figures 3.4a, 3.4b, 3.4c, 3.4d, 3.4e respectively). This large gap is attributed to the fact that the later approach does not propose any mechanism to filter out any possible selfish device in the client selection process. This causes the possibility of selection of a large number of selfish devices to participate in the FL training.

3.6 Conclusion

In this chapter, we introduced a *Coalitional Federated Learning (CFL)* framework that redefines how trust and collaboration are established in federated environments. Unlike conventional approaches where the server unilaterally governs trust and participation, our framework decentralizes this process by introducing a client-to-client trust mechanism. This enables clients to evaluate one another based on both objective resource usage and subjective recommendations, resulting in a more democratic ecosystem.

By incorporating peer-derived trust into coalition formation, clients gain the autonomy to choose whom to collaborate with, forming trustworthy communities that exclude selfish participants. This design not only reduces the bias coming from a single, server-centric viewpoint but also empowers honest devices that are often overlooked by global policies to play a more active role in the learning process. Through the proposed hedonic coalition formation mechanism, devices self-organize into stable and trustworthy groups, while selfish clients are naturally isolated and excluded from participation.

Ultimately, our framework shifts federated learning toward a more trustworthy paradigm, where trust is collectively established, and collaboration is maximized through reputation and coalition stability. Experimental results demonstrate that this decentralized trust model lowers the prevalence of selfish devices and improves the global model accuracy compared to other methods.

CHAPTER 4 ARTICLE 2: TOWARDS VOX POPULI IN FEDERATED LEARNING: A FAIR AND INCLUSIVE CLIENT SELECTION FRAMEWORK

Accepted for publication in IEEE Transactions on Artificial Intelligence on September 10, 2025. Authors: Sarhad Arisdakessian, Osama Wehbi, Omar Abdul Wahab, Azzam Mourad, Hadi Otrok.

In this chapter, we propose another client selection approach through the lens of Stackelberg games. Mainly, current FL frameworks face significant challenges in fair client selection, especially when relying on traditional Stackelberg game models [59, 60, 62]. While Stackelberg games are widely used to optimize hierarchical interactions, most existing approaches treat the central server as the leader, with clients as followers, resulting in a rigid, centralized client selection that overlooks resource disparities among clients. Such a setup limits the participation of clients with fewer resources, as it inherently favors those with higher computational and data capabilities, leading to biased learning outcomes and reduced data diversity. Additionally, these approaches lack a robust metric for evaluating the data quality and representativeness of each client's contribution, which is crucial in heterogeneous FL environments. To address this need, an effective system must incorporate a quantitative *data score* metric to measure each client's data value, factoring in elements such as class diversity, dataset size, and distribution balance. Without this, understanding each client's data contribution remains challenging, limiting the FL system's ability to optimize learning.

Furthermore, conventional Stackelberg-based methods [68, 69] lack mechanisms for fostering collaborative, decentralized coalitions among clients. This absence prevents the formation of dynamic, adaptable groups within the client network that could pool resources and leverage unique data types to enhance representation. As a result, clients with lower resources have limited access to the benefits of FL, and opportunities for diverse, coalition-driven learning environments are missed.

Given these limitations, we believe there is a clear need for a comprehensive strategy that enables a fair, balanced client selection process, promoting inclusivity for all participating clients.

4.1 Client Categorization and Data Evaluation

In this section, we describe the two key steps that lay the foundation for our proposed framework. First, we explain the process of clustering clients into leaders and followers based on their resources. Then, we introduce our data scoring mechanism that quantitatively assesses each client's data quality and representativeness.

4.1.1 Clustering: Leaders and Followers

Our framework begins with a clustering phase, where devices are classified as leaders or followers based on their resource capabilities. Leaders subsequently form coalitions by selecting followers through a Stackelberg game-based mechanism, wherein followers compete by offering payments to maximize their likelihood of being included in the coalition. Aggregated data scores are computed to assess the combined utility of leader-follower coalitions. Leaders then optimize their utility by strategically balancing data diversity and follower contributions. The framework is designed to ensure scalability, fairness, and computational feasibility, making it well-suited for large-scale FL environments. A flowchart illustrating the workflow of our proposed StackFed framework is provided in Figure 4.1.

Hereafter, we focus on the clustering phase, which serves as the initial step and entry point of our framework. Each client is equipped with various system-based resources such as Central Processing Unit (CPU), Random Access Memory (RAM), and bandwidth (BW), as well as data-based resources, such as Data Classes (DC), Data Volume (DV), and Data Distribution (DD).

We consider time (T) $t = 0$ the initialization time for our FL environment equipped with a server, and $I = \{1, 2, 3, \ldots, n\}$ a set of clients that have joined the network. At this phase, we are interested in categorizing different clients as leaders or followers. Therefore, our end-goal is to convert the set I into distinct sets of leaders $L = \{l_1, l_2, l_3, \ldots, l_n\}$ and followers $F = \{f_1, f_2, f_3, \ldots, f_n\}$.

Algorithm 6 highlights the process for categorizing clients into leaders and followers. It takes a set of all clients I as input, and outputs two disjoint sets: L for leaders and F for followers (Lines 1-2). The process begins with each client sending its resource metadata, consolidated into a feature vector ($\mathbf{x}_i$), to the server (Lines 3-6). After receiving all feature vectors, the server performs a global Min-Max normalization on the entire dataset (Lines 7-10). This crucial step creates a fair and consistent scale for all client resources. The server then initializes empty sets to be populated with leaders and followers (Line 11). Next, the server runs the DBScan clustering algorithm on the normalized feature vectors to partition the clients into a set of clusters ($\mathcal{K}$), where each cluster contains clients with similar resource profiles (Line 12). To identify the leader coalition, the server first calculates the centroid (μ_j) for each cluster (Lines 13-15). It then selects the single leader cluster, K_L, by formally identifying the cluster whose centroid has the maximum Euclidean norm (Line 16). Finally,

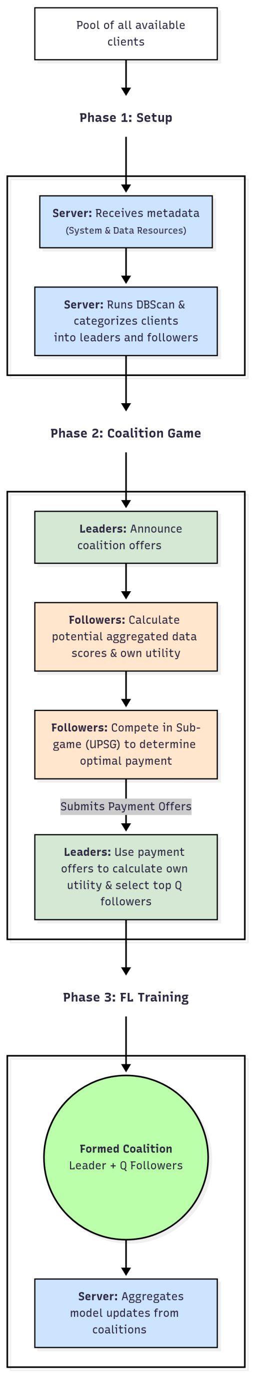

Pool of all available clients
Phase 1: Setup
Server: Receives metadata (System & Data Resources)
Server: Runs DBScan & categorizes clients into leaders and followers
Phase 2: Coalition Game
Leaders: Announce coalition offers
Followers: Calculate potential aggregated data scores & own utility
Followers: Compete in Sub-game (UPSG) to determine optimal payment
Submits Payment Offers
Leaders: Use payment offers to calculate own utility & select top Q followers
Phase 3: FL Training
Formed Coalition Leader + Q Followers
Server: Aggregates model updates from coalitions

Figure 4.1 A flowchart illustrating the proposed StackFed framework.

Algorithm 6 Clustering: Leaders & Followers

```
1: **Input:** A set of clients $I$.
2: **Output:** A set of leaders $L$ and a set of followers $F$.

3: for each client $i \in I$ do
4:     Let $\mathbf{x}_i$ be the feature vector $(CPU_i, RAM_i, BW_i, DC_i, DV_i, DD_i)$.
5:     Send $\mathbf{x}_i$ to the central server.
6: end for

7: Let $X = \{\mathbf{x}_i\}_{i \in I}$ be the set of all received feature vectors.
8: for each vector $\mathbf{x}_i \in X$ do
9:     Compute Min-Max normalized feature vector $\mathbf{x}'_i$
10: end for

11: Initialize $L \leftarrow \emptyset$, $F \leftarrow \emptyset$

12: Run DBScan on the clients in $I$ using their normalized feature vectors, $\mathbf{x}'_i$, to produce a
    set of clusters $\mathcal{K} = \{K_1, K_2, ..., K_m\}$.

13: for each cluster $K_j \in \mathcal{K}$ do
14:     Compute the centroid $\mu_j = \frac{1}{|K_j|} \sum_{i \in K_j} \mathbf{x}'_i$.
15: end for
16: $K_L \leftarrow \underset{K_j \in \mathcal{K}}{\mathrm{argmax}} ||\mu_j||_2$
17: $L \leftarrow K_L$
18: $F \leftarrow I \setminus L$
19: Return $L, F$
```

the leader set L is assigned all clients from the winning cluster (Line 17), and the follower set F is defined as all other clients from the original set I (Line 18). The algorithm concludes by returning these two distinct sets (Line 19).

4.1.2 Data Scores

Data heterogeneity can manifest in various forms. For instance, some clients may have a large amount of data restricted to certain classes, whereas others may hold smaller datasets covering a broader range of classes. Additionally, some clients may display mixed scenarios with varied distributions. In this section, we outline the quantification of a score that reflects these characteristics in two scenarios: individually for a single client, and for the aggregated data score when a follower client joins a leader in pooling their data.

Individual Data Scores

To accomplish this, we utilize three distinct metrics to quantify a client's *Data Score* (DS) through a weighted statistical approach.

Data Classes (DC_i)

This metric represents the proportion of classes a device holds relative to the total classes, and is calculated as

$$DC_i = \frac{|\mathcal{C}_i|}{|\bigcup_{j \in I} \mathcal{C}_j|}, \tag{4.1}$$

where $\mathcal{C}_i$ is the set of unique classes on client i, I is the set of all clients in the system, and the denominator represents the cardinality of the union of all class sets, and $|\cdot|$ denotes the cardinality of the set. For example, in an MNIST task, if a device has data for digits 3, 5, and 8, its DC_i score is 0.3, as it has 3 out of 10 possible classes.

Data Volume (DV_i)

This metric represents the amount of data a client holds relative to the total data across all clients, providing a balanced and fair assessment, and is calculated using

$$DV_i = \frac{N_i}{\sum_{j \in I} N_j}, \tag{4.2}$$

where N_i is the number of data samples on client i, and $\sum_{j \in I} N_j$ is the total number of samples summed over all clients j in the set I.

Data Distribution (DD_i)

This metric assesses the balance of data across classes for each client. A higher DV_i score doesn't ensure that data is evenly distributed among classes, making DD_i essential for fair representation. We use the Shannon diversity index [101], which measures entropy, originally applied in ecology to estimate species diversity by considering both species variety and abundance. Here, we draw an analogy between species and data classes, with their quantities as abundance, applying Shannon's formula to gauge data distribution balance across devices. This score is calculated using

$$DD_i = -\sum_{c=1}^{|\mathcal{C}_{\text{total}}|} P_c \log P_c, \tag{4.3}$$

where $|\mathcal{C}_{\text{total}}|$ is the total number of unique classes in the system, and P_c is the proportion of data points on client i that belong to class c.

Note that each client performs these calculations locally. Accordingly, the Data Score DS_i of a client i is defined as:

$$DS_i = DC_i \times DV_i \times DD_i. \tag{4.4}$$

To provide additional flexibility, we make our formula weighted, allowing task owners to prioritize specific aspects as needed. This enables them to assign higher weights to certain metrics based on the context or application, while reducing weights on others. By incorporating these weights into Equation 4.4, we obtain the weighted version, where: W_{DC}, W_{DV}, and W_{DD} are the weights assigned for the data class, data volume and data distribution components respectively. Accordingly, the equation is represented as:

$$DS_i = (W_{DC} \cdot DC_i) \times (W_{DV} \cdot DV_i) \times (W_{DD} \cdot DD_i). \tag{4.5}$$

With the individual scoring mechanism established, we now describe how these metrics are aggregated when a leader and follower collaborate to form a coalition.

Aggregated Data Scores

Effective collaboration between leaders and followers is essential for maximizing data diversity, and reducing bias. When a follower joins a leader's coalition, the aggregation of their data provides enriched data resources, fostering a more balanced and comprehensive dataset. This section describes the specific aggregation functions that govern how leaders and followers work together, quantify their combined contributions, and support the development of a representative learning model.

Aggregated Data Classes DC_{l_j,f_i}

This metric represents the combined set of classes when a leader l_j and a follower f_i collaborate, accounting for any overlap in classes, and is calculated using

$$DC_{l_j,f_i} = \frac{|\mathcal{C}_{l_j} \cup \mathcal{C}_{f_i}|}{|\mathcal{C}_{\text{total}}|}, \tag{4.6}$$

where $\mathcal{C}_{l_j}$ is the set of unique classes held by leader l_j, and $\mathcal{C}_{f_i}$ is the set of unique classes held by follower f_i, and $|\mathcal{C}_{\text{total}}|$ is the total number of unique classes.

Aggregated Data Volume DV_{l_j,f_i}

This value represents the aggregated quantity of data for both clients after collaboration, and is calculated using

$$DV_{l_j,f_i} = \frac{N_{l_j} + N_{f_i}}{\sum_{j \in I} N_j}, \tag{4.7}$$

where N_{l_j} and N_{f_i} are the number of data samples on the leader and follower respectively, and $\sum_{j \in I} N_j$ is the denominator representing the total number of samples.

Aggregated Data Distribution DD_{l_j,f_i}

This value represents the new Shannon distribution after aggregating the classes and their respective quantities with the collaboration of two clients l_j, f_i, and is calculated using

$$DD_{l_j,f_i} = -\sum_{c=1}^{|\mathcal{C}_{l_j} \cup \mathcal{C}_{f_i}|} P_c \log P_c, \tag{4.8}$$

where the summation is over all unique classes c in the coalition set $\mathcal{C}_{l_j} \cup \mathcal{C}_{f_i}$, and P_c is the proportion of data points belonging to class c in the combined dataset of the leader and follower.

Finally, the final aggregated data score between a leader and follower would be represented as:

$$DS_{l_j,f_i} = (W_{DC} \cdot DC_{l_j,f_i}) \times (W_{DV} \cdot DV_{l_j,f_i}) \times (W_{DD} \cdot DD_{l_j,f_i}). \tag{4.9}$$

This aggregated score is a critical component of our framework, as it quantifies the value a leader-follower coalition brings to the learning task.

4.2 Stackelberg Game Theoretical Framework

In this section, we introduce our novel game-theoretic framework for fair client selection. We begin by formulating a Stackelberg game that underpins our decentralized approach, followed by detailed definitions of the utility functions for both leaders and followers. We then describe the follower's payment selection process and present our strategy for determining their optimum payment offers. Next, we derive the Nash equilibrium of the game, and finally, we discuss how leaders optimize their utility based on these interactions. Together, these components form a comprehensive strategy that overcomes the limitations of traditional, server-centric approaches.

4.2.1 Game Theoretical Analysis

In this section, we analyze the proposed Stackelberg game, outlining its motivation, structure, utility definitions, follower sub-game, and equilibrium analysis.

Stackelberg games [102] model hierarchical interactions between leaders (L) and followers (F). In our framework, leaders, act first by offering coalition opportunities. Followers, less likely to be selected independently, strategically choose to join leaders to access the global model's benefits.

While one might assume that selecting only leaders suffices post-clustering, this is suboptimal. Leaders may lack certain data classes, exhibit imbalanced distributions, or have insufficient volumes; limitations that followers can help address. Ignoring followers restricts data diversity and fairness, and diminishes the model's convergence and performance. Thus, enabling collaboration between leaders and followers ensures a more balanced, inclusive, and effective learning process. Our use of Stackelberg games facilitates such cooperation by allowing coalition formation. Leaders maintain their strategic advantage, while followers gain meaningful participation opportunities. A coalition, led by a resourceful client, operates as a single abstracted entity; concealing internal dynamics from the server and task owner. This abstraction ensures scalability and simplifies integration into standard FL protocols while preserving fairness and diversity.

Formally, a coalition $\mathcal{K}_i$ is led by a leader device L_i. Each coalition consists of the leader and a set of followers, denoted by F_i. The structure of the coalition is therefore represented as:

$$\mathcal{K}_i = \{L_i\} \cup F_i. \tag{4.10}$$

Technically, leveraging its superior resources and strategic advantages, the leader has the privilege to pre-select a subset of followers for potential collaboration (e.g., the top 10 clients). From this subset, the leader further establishes a quota for final selection (e.g., choosing 5 clients out of the initial 10). Leaders then publish their offers, detailing attributes such as data resources, to the pre-selected followers. On their end, followers then engage in a competitive sub-game, determining the payments they will propose to the leader in an effort to maximize their utility by securing a position in the final selection without excessive expenditure. The leader's objective is to select the optimal subset of followers that maximizes its utility, balancing data enhancement with revenue generation. This two-stage Stackelberg game model enables the leader to optimize its strategy by anticipating and incorporating the best responses from followers.

By enabling follower devices to join a coalition, we introduce an essential and often overlooked element of fairness to the FL process, giving resource-limited devices the opportunity to participate, contribute valuable data, and benefit from the global model. This approach diverges from most literature, where resource constraints often exclude less-resourceful devices from selection, thus limiting the diversity and inclusivity of the data pool. Our framework not only democratizes client participation but also enhances model robustness by integrating a wider range of data sources. We employ backward induction to analyze the game's equilibrium, deriving the optimal payment strategies that followers are likely to adopt in response to the leader's offer, thereby integrating fairness with utility maximization for the leader.

4.2.2 Utility Functions

In our game, payoffs in the forms of payments are essential, with leaders expecting compensation from followers that directly impacts their selection preferences. Unlike the majority of existing literature, where leaders incentivize followers through payments, our approach distinctively positions followers as the paying side, compensating leaders instead. This model introduces a novel perspective, wherein followers offering higher payments have a greater likelihood of selection, thus benefiting the leader economically and fostering a fair competitive environment among followers. It is worth noting that our methodology for defining the utility functions is inspired by [103].

Follower's Utility

Given their limited resources, followers must collaborate with leaders to improve their chances of accessing the global model. The utility function for followers includes two main components: the change in their data score after aggregation with the leader, and the proportion of their payment relative to other followers' payments. This relative payment serves two purposes: (1) it makes a follower's utility proportional to its contribution compared to all followers, encouraging competitive payments, and (2) it incentivizes followers to increase their payment to enhance their collaboration prospects and utility. Finally, we subtract the follower's payment to account for incurred costs. The resulting utility function for followers is defined as

$$U_{f_i} = \frac{P_{f_i}}{\sum_{f \in F} P_f} [\Delta \left(DS_{f_i,l_j} \right)] - P_{f_i}, \tag{4.11}$$

where P_{f_i} is the payment made by follower f_i to leader l_j, $\sum_{f \in F} P_f$ denotes the total sum of payments made by all followers in the pre-selected set of followers and $\Delta \left(DS_{f_i,l_j} \right)$ is the variation of the data score for the follower f_i after collaborating with the leader l_j, and $\Delta(DS_{f_i,l_j})$ is defined as

$$\Delta(DS_{f_i,l_j}) = DS_{f_i,l_j} - DS_{f_i}, \tag{4.12}$$

where DS_{f_i,l_j} denotes the data score for follower f_i after aggregation with leader l_j, and DS_{f_i} is the data score for follower f_i before aggregating with leader l_j.

This utility function explicitly models the follower's rational trade-off. The first term, $\frac{P_{f_i}}{\sum_{f \in F} P_f}[\Delta(DS_{f_i,l_j})]$, captures the expected benefit of being selected, reflecting the proportional influence of the follower's payment on coalition formation, scaled by the value of their data. The second term, $-P_{f_i}$, reflects the direct monetary cost of the payment. Maximizing this utility guides the follower toward an optimal payment that balances increased selection probability with cost minimization.

Leader's Utility

The leader's utility is the sum of the benefits gained from each follower selected to be in its coalition. The leader's utility is then defined as

$$U_{l_j} = \sum_{f \in F^L} \Big(\Delta(DS_{l_j,f}) \times P_f \Big) \tag{4.13}$$

where F^L is the set of followers selected by the leader (the output of Algorithm 3). The sum is taken over all followers $f \in F^L$, where P_f is the payment from follower f, and $\Delta(DS_{l_j,f})$ is the change in the leader's data score from collaborating with that follower, defined as:

$$\Delta(DS_{l_j,f}) = DS_{l_j,f} - DS_{l_j} \tag{4.14}$$

where $DS_{l_j,f}$ denotes the data score for leader l_j after aggregation with follower f, DS_{l_j} is the data score for leader l_j before aggregating with follower f. The leader's utility formulation emphasizes that selection is not based solely on monetary offers. Specifically, the utility function considers both the payment offered by each follower and the value of their data. This design ensures that a follower offering relatively lesser payment but with highly complementary data may yield greater utility than one offering a large payment but with redundant data. This dual-factor preference captures both monetary and non-monetary incentives in a unified formulation.

4.2.3 Follower's Payment Selection Game

To improve their chances, followers can join a coalition led by a resourceful leader, yet they must compete with other followers for these limited spots. Given this competitive setting, followers need to determine optimal payment offers to appeal to the leader without compromising their own utility. This scenario creates the need for an uncooperative sub-game in the main Stackelberg game among the followers, where each follower seeks to maximize its utility while considering the actions of others. However, each follower must carefully consider its payment offer, as overpaying without a proportional improvement in data score could lead to a net utility loss. Accordingly, each follower f_i aims to determine an optimal payment P_{f_i}. This sub-game, termed the Uncooperative Payment Selection Game (UPSG), is a non-cooperative, static and one-shot game where all followers decide on a payment offer. It is assumed that leaders are honest and reliable and broadcast their relevant data score metadata along with their offers, enabling followers to calculate their potential data score improvement. The initial payment (IP_{f_i}), defined in Equation 4.15, represents the data score improvement that a follower f_i can achieve by collaborating with leader l_j and forms the basis of the game:

$$\mathrm{IP}_{f_i} = \Delta(DS_{f_i,l_j}). \tag{4.15}$$

After calculating their initial payment amounts, followers broadcast these values to other followers competing for membership in the leader's coalition. It is also assumed the followers are honest. The strategy space for each follower is the set of all possible payments, defined as a continuous interval between a minimum of zero and a maximum budget, $\overline{P}$. With this shared information, each follower can assess how its offer compares to others and engage in the Uncooperative game to select the optimal payment. The specific method by which clients communicate their payment information is beyond the scope of this work. One possible scenario is, given that clients have already established communication with the central server upon joining the network, they can utilize this connection to relay payment information. It is assumed that the server acts as an honest intermediary, facilitating secure message exchange between clients. By passing the messages through the server, clients can effectively share necessary information with other followers without requiring direct peer-to-peer communication, thus maintaining both scalability and simplicity in message transmission.

The non-cooperative Payment Selection Game (UPSG) is defined by the 3-tuple

$$\text{UPSG} = \langle F, \{P_f\}_{f \in F}, \{U_f\}_{f \in F} \rangle, \tag{4.16}$$

where F is the set of players (the followers), $\{P_f\}_{f \in F}$ is the set of action profiles, where each follower $f \in F$ chooses a payment strategy $P_f \in [0, \overline{P}]$, and $\{U_f\}_{f \in F}$ is the set of utility functions. This game is played in a non-cooperative setting, where each follower aims to choose a strategy that maximizes their utility $U_f(P_f, P_{-f})$, which depends on their own strategy and the strategies of all other followers, denoted by P_{-f}. The Nash equilibrium of this game denotes the state where no follower can improve its utility by unilaterally changing its strategy.

4.2.4 Nash Equilibrium

The Nash equilibrium is reached when each follower finds the best payment strategy it needs to pay to the leader as a response to the leader's offer vis-à-vis the other follower's payment strategies P_{-f}. This implies that for any client, its final strategy is the highest possible utility score for that individual client, such that, any other strategy for the client would not yield a higher utility score. When all clients reach this stage, we state that the Nash Equilibrium has been established and no follower has intuition to deviate from its current strategy.

A payment vector $P = (P_{f_1}, P_{f_2}, \ldots, P_{f_n})$ is a Nash equilibrium of UPSG if, for every follower $f_i \in F$, letting $P_{-f_i} = (P_{f_1}, \ldots, P_{f_{i-1}}, P_{f_{i+1}}, \ldots, P_{f_n})$ denote the payments of all other followers, and letting $(P'_{f_i}, P_{-f_i}) = (P_{f_1}, \ldots, P_{f_{i-1}}, P'_{f_i}, P_{f_{i+1}}, \ldots, P_{f_n})$ denote the payment profile in which follower f_i unilaterally deviates to some alternative payment $P'_{f_i} \in [0, \overline{P}]$, the following condition holds:

$$U_{f_i}(P_{f_i}, P_{-f_i}) \geq U_{f_i}(P'_{f_i}, P_{-f_i}), \quad \forall P'_{f_i} \in [0, \overline{P}].$$

Accordingly, the strategy space for the payment can be defined as

$$P = [P_F]_{\forall f \in F} : 0 \leq P_f \leq \bar{P}, \tag{4.17}$$

where $\bar{P}$ is the maximum amount of the payment that can be paid.

Since the utility function of a follower f_i (Eq. 4.11) is continuous in P_{f_i}, derivatives can be used to find the best response function. The Nash equilibrium is determined by identifying the optimal strategy (i.e., payment) each follower should employ in response to the leader's offer and the strategies of other followers, using a best-response approach.

We can observe that the second order derivative of U_{f_i} against P_{f_i} is always negative, which implies that U_{f_i} is downwards concave in P_{f_i}.

Theorem 1: $U_f = \frac{P_f}{\sum_{f \in F} P_f}[\Delta\ (DS_{f,l})] - P_f$ is concave down in P_f.

From this, we can deduce that P_{f_i} is optimum for U_{f_i} when $\frac{\partial U_{f_i}}{\partial P_{f_i}} = 0$.

Theorem 2: The equilibrium of the game is reached via:

$$P^*_{f_i} = \sqrt{\left[\Delta(DS_{l_j,f_i}) \sum_{f \in F \setminus \{f_i\}} P_f\right]} - \sum_{f \in F \setminus \{f_i\}} P_f. \tag{4.18}$$

To ensure that the payment remains within the feasible range $[0, \bar{P}]$, we define the final equilibrium payment as

$$P^{eq}_{f_i} = \begin{cases} 0, & \text{if } P^*_{f_i} < 0 \\ P^*_{f_i}, & \text{if } 0 \leq P^*_{f_i} \leq \bar{P} \\ \bar{P}, & \text{if } P^*_{f_i} > \bar{P}, \end{cases} \tag{4.19}$$

where $P^*_{f_i}$ is the unconstrained optimal payment defined in Equation 4.18. Equation 4.19 applies the necessary boundary constraints to produce the final, feasible equilibrium payment.

4.2.5 Follower's Optimum Payment Selection Algorithm

In this subsection, we introduce Algorithm 7, which outlines the optimal payment selection scheme for followers. In this approach, each follower determines an optimal price it is willing to pay the leader to increase its likelihood of being selected.

Algorithm 7 Follower's Payment Selection Algorithm

```
1: Input: Set of followers $F$, leader's resource metadata
2: Output: Optimum payments $P_f^{eq}$ for all $f \in F$
3:
4: for each $f \in F$ do
5:     Compute $\Delta(DS_{f,l})$ as per Eq. 4.12
6:     Compute $\mathrm{IP}_f$ as per Eq. 4.15
7: end for
8:
9: $S \leftarrow \sum_{f \in F} \mathrm{IP}_f$
10:
11: for each $f \in F$ do
12:     Compute the sum $S - \mathrm{IP}_f$
13:     Compute $P_f^{eq}$ using this value as per Eq. 4.19
14: end for
```

Accordingly, algorithm 7 highlights the follower's payment selection procedure. It begins by taking as input the set of followers F and the leader's resource metadata (lines 1–2). For each follower f in F, the algorithm computes the change in data score, $\Delta(DS_{f,l})$, as defined in Eq. 4.12, and then determines the initial payment IP_f according to Eq. 4.15 (lines 4–7). Once all initial payments are computed, their total sum S is calculated (line 9). Next, for each follower, the algorithm computes the sum $S - \mathrm{IP}_f$, representing the total of all initial payments except that of f, and uses this value to calculate the optimum payment P_f^{eq} via the equilibrium formula in Eq. 4.19. The final output consists of the optimum payments for all followers (lines 11–14). In the payment selection phase, each follower computes its optimal payment by evaluating the relevant equilibrium conditions. The algorithm achieves a total computational complexity of $\mathcal{O}(|F|)$, where $|F|$ is the number of followers. This linear complexity is achieved by precomputing the total sum of initial payments and then, for each follower, obtaining the required sum over all other followers in constant time by subtraction, given both initial and equilibrium payment computations require only a single pass through the followers.

4.2.6 Leader's Utility Optimization

The leader's objective is to maximize its utility. This means the leader will benefit both, from complementary data points, as well as payments incurred from the followers. Because of this, it is to the leader's own benefit to select an optimal quota (Q) of followers to collaborate with which will accordingly increase its utility. Based on this, we can use the optimal payment's equation (4.18) inside the leader's utility function U_l (4.13) to have:

$$U_{l_j} = \sum_{f \in F^L} \Big(\Delta(DS_{l_j,f}) \times P_f^{eq} \Big) \tag{4.20}$$

We propose Algorithm 8 for the leader's utility optimization algorithm, in which the leader selects a set of followers which maximize its utility. Given that our approach employs data scores to quantify each client's potential contribution before the FL process begins, the leader can form a coalition with the best set of clients that maximizes its utility. This strategy mirrors coalition-based FL scenario, where dynamically altering the coalition every round would incur significant computational and time costs, making it impractical. By fixing the coalition at the outset, our framework guarantees that the global model is consistently trained on a stable set of data sources; which are already computed a priori to be the best combination in the current pool of clients for the task; reducing update variability. Additionally, static selection minimizes the overhead associated with frequent re-clustering and re-selection, ensuring that resource-limited clients remain consistently integrated. Newly joined clients, or clients with updated data points can participate in the next task where the process repeats. Our framework assumes that rational followers, having already invested through payments, are economically disincentivized from withdrawing mid-task. This alignment of incentives promotes stable coalition participation for the full duration of the task, as participants have no strategic benefit in leaving prematurely. Unexpected dropouts due to system-level issues; such as battery depletion or network failure; are treated as non-strategic events. While our current design focuses on stability, addressing such operational failures through fault-tolerant mechanisms (e.g., redundancy or participant replacement) is left as a direction for future work. Furthermore, while optimal exhaustive solutions provide the theoretical advantage of identifying the absolute best configuration for leader-follower coalitions, they become computationally impractical as the system scales. The exhaustive search requires evaluating all possible combinations of followers, leading to a computational complexity of $O(n^Q)$, where n is the number of potential followers and Q is the coalition size. This exponential growth in complexity quickly renders such approaches infeasible in real-world scenarios involving large-scale federated systems. On the other hand, greedy-heuristic approaches offer a practical alternative by iteratively selecting followers based on their immediate contributions to the leader's utility, such as data score improvement and payment.

Algorithm 8 Leader's Utility Optimization Algorithm

```
1: Input: Maximum quota of followers per coalition $Q$
2: Input: Set of possible followers $F$
3: Input: Follower's payments $P_f^{eq}$ from Alg. 7
4: Output: Set of followers $F^L$ confederating with leader
5:
6: Sort followers $f \in F$ by weighted contribution
7:     $w_f = \alpha \cdot \Delta(DS_{l,f}) + \beta \cdot P_f^{eq}$ in descending order
8: Select the top $Q$ followers from $F$ and assign to $F^L$
9: Calculate $U_l(k)$ for the selected followers as per Eq. 4.20

10: Return $F^L$
```

Therefore, Alg. 8 is designed using greedy heuristic to efficiently determine the optimal subset of followers (F^L) for a leader's coalition, emphasizing computational scalability and practical applicability. The algorithm begins by ranking the followers based on a weighted contribution score (w), calculated as $w = \alpha \cdot \Delta(DS_{l,f}) + \beta \cdot P_f^{eq}$, where $\Delta(DS_{l,f})$ represents the data score improvement contributed by a follower, P_f^{eq} is the payment offered, and α, β are weights that allow the leader to prioritize either data contributions or monetary incentives. Sorting the followers by their weighted scores ensures that those offering the highest combined contributions are prioritized. From the sorted list, the algorithm selects the top Q followers, adhering to the leader's quota and focusing on the most beneficial followers. It then calculates leader's utility $U_l(k)$, which integrates both data improvements and payments as a reporting step. Finally, the algorithm returns the set of followers (F^L) that form the coalition.

This greedy heuristic approach avoids the computational complexity of exhaustive searches $O(n^Q)$, reducing the runtime to $O(n \log n)$ due to the sorting step. While it does not guarantee a globally optimal solution, the algorithm provides a practical, near-optimal alternative that aligns with real-world constraints, enabling efficient coalition formation even in large-scale FL environments. By incorporating the weighted contribution score, the algorithm remains flexible, allowing leaders to adapt their selection strategies based on contextual priorities, such as enhancing data diversity or maximizing monetary incentives.

Computational Analysis & Scalability

To demonstrate the feasibility and scalability of our approach, we analyze and discuss the computational complexity and overhead of each stage in the proposed framework. The goal is to showcase its applicability in large-scale real-world FL systems.

Stage 1: Clustering of Leaders and Followers
Overall Complexity: $O(n \log n)$

The clustering stage, initializes the system by dividing devices into leaders and followers. Each device shares its metadata, including CPU, RAM, bandwidth, and data characteristics, with the central server. This process scales linearly with the number of devices n, resulting in a complexity of $O(n)$. The normalization phase also has a complexity of $O(n)$. Then DBScan is employed to cluster devices based on their metadata. Its complexity depends on the implementation but with spatial indexing, DBScan achieves near-linear complexity $O(n \log n)$. Devices in the cluster with the highest resource values are designated as leaders, while others become followers. This step involves comparisons and indexing, scaling as $O(n)$. Thus, the overall complexity of this stage can be characterized as $O(n \log n)$.

Stage 2: Data Score Computation
Overall Complexity: $O(k)$
Each device computes its data score locally using three metrics: data classes, data volume, and data distribution. For data classes, the complexity is $O(k)$, where k is the number of data classes. For data volume, given it is a simple ratio calculation, the complexity is of $O(1)$, while the data distribution that relies on Shannon entropy computation has a complexity of $O(k)$. Combining metrics with weights has a complexity of $O(1)$. Thus the overall complexity is $O(k)$. Since computation occurs in parallel across devices, the parallel runtime remains $O(k)$, independent of the number of devices n.

Stage 3: Aggregated Data Scores
Overall Complexity: $O(Q \cdot k)$
When followers join a leader's coalition, aggregated data scores are computed. For the aggregated data classes, aggregating class distributions across the coalition has a complexity of $O(k)$, while aggregated data volume has a complexity of $O(1)$, and aggregated data distribution using Shannon's entropy has a complexity of $O(k)$. Thus, the overall complexity for each coalition is $O(k)$. Accordingly, with Q followers, the total complexity becomes $O(Q \cdot k)$. This remains efficient for typical k and moderate Q.

Stage 4: Follower's Payment Selection Game
Overall Complexity: $O(Q)$

The algorithm runs in linear time with a complexity of O(|F|), where |F| is the number of followers. This efficiency is achieved because the total sum of initial payments is calculated in a single pass, allowing the sum of all other followers' payments to be determined in constant time for each follower through simple subtraction.

Stage 5: Leader's Utility Optimization
Overall Complexity: $O(Q \log Q)$
The leader selects the subset of followers for a leader's coalition. Sorting Q followers based on weighted contributions scales as $O(Q \log Q)$. Thus the overall complexity for each leader given a preselected quota Q of followers, is $O(Q \log Q)$, ensuring scalability for large systems.

System-Wide Complexity:
Combining all stages, for bounded k and Q the system-wide complexity is dominated by clustering and leader-follower interactions $O(n \log n)$. The framework's reliance on parallel processing significantly enhances real-world feasibility, as operations for leaders and followers occur independently, distributing computational load and minimizing bottlenecks. Localized computations, such as data score calculations, further reduce server-side overhead, allowing devices to process data simultaneously.

Scalability Discussion

Our design shifts most computations to the client side and scopes interactions locally within leader–follower coalitions. Leader selection via DBScan scales as $O(n \log n)$, and each leader only interacts with a small subset Q of followers ($Q \ll n$). Furthermore, data score computations and payment strategies are executed in parallel across clients. Importantly, most of these computations; such as leader clustering, data score evaluation, and optimal payment derivation; are performed as batch processes prior to training and do not occur during runtime model updates. As a result, one process does not hinder another, and the training rounds remain efficient and minimally delayed. This modular and decentralized design ensures that even as the total number of clients n increases significantly, the per-device overhead remains bounded, and the server is not a bottleneck. Overall, our framework demonstrates strong scalability characteristics suitable for real-world federated systems involving hundreds or thousands of clients.

4.3 Simulations and Experiments

We evaluate our proposed framework, StackFed, through simulations implemented in Python 3 using PyTorch, and Torchvision. Our setup simulates FL environments with client heterogeneity in both system and data resources.

For our experiments, we use three benchmark datasets: FashionMNIST [1], K-MNIST [2], and USPS [3]. FashionMNIST provides visual complexity, K-MNIST introduces challenging class distributions, and USPS offers a dataset with distinct non-IID characteristics. These datasets were selected to cover a range of difficulty levels and distribution types commonly encountered in FL. We also vary the coalition size Q, which determines the number of followers per leader, to study its impact on accuracy, fairness, and efficiency.

To assess the performance of StackFed, we compare it against four baselines, each representing a different selection strategy. The first is the Random baseline [1], which randomly selects clients for training, serving as a neutral benchmark with no preference for data or system attributes. The second baseline, Only Leaders, selects clients purely based on resource quality, reflecting resource-driven policies commonly found in traditional FL settings. We also include two state-of-the-art (SOTA) approaches for deeper comparison, selected for their thematic alignment with our proposed StackFed framework. The first SOTA baseline [50] denoted as $\text{SOTA}_{\text{multi}}$, is a multi-criteria client selection scheme that, like StackFed, considers both system and data resources. On the system side, it evaluates CPU, memory to ensure clients can meet a server-defined training deadline. On the data side, it prioritizes data class diversity to mitigate imbalance, conceptually similar to our data score metric, which captures class diversity, volume, and distribution. The second SOTA baseline [66], $\text{SOTA}_{\text{stack}}$, employs Stackelberg game, but follows a conventional server-centric approach in which the server acts as a single leader broadcasting a global reward to incentivize client participation. In contrast, StackFed decentralizes the game by enabling resource-rich clients to assume the role of leaders who are directly compensated by resource-constrained followers in exchange for coalition membership. This peer-level leader-follower interaction differentiates StackFed through its support for fair, scalable, and coalition formation.

Furthermore, our evaluation focuses on, computational complexity & scalability, accuracy, class imbalance, and privacy. These experiments demonstrate that StackFed enhances model performance while promoting fairness, inclusivity, and practical deployment feasibility in FL.

Our empirical results, validate the assumptions of the proposed game-theoretic model. This

[1]`https://github.com/zalandoresearch/fashion-mnist`
[2]`https://github.com/rois-codh/kmnist`
[3]`https://git-disl.github.io/GTDLBench/datasets/usps_dataset/`

is because observed behavior confirms that rational followers are incentivized to contribute competitively and that leaders select coalitions that enhance overall data quality and diversity.

4.3.1 Computational Complexity & Efficiency

A critical aspect that ensures the practicality of our approach in real-world scenarios, despite its modular structure and multiple components, is its computational efficiency. In this initial set of experiments, we focus on validating this efficiency by comparing the performance of our greedy heuristic approach against the exhaustive search method. Through this comparison, we demonstrate how our approach significantly reduces computational overhead, making it well-suited for large-scale FL environments while keeping the utility outcome of both approaches identical, or very close.

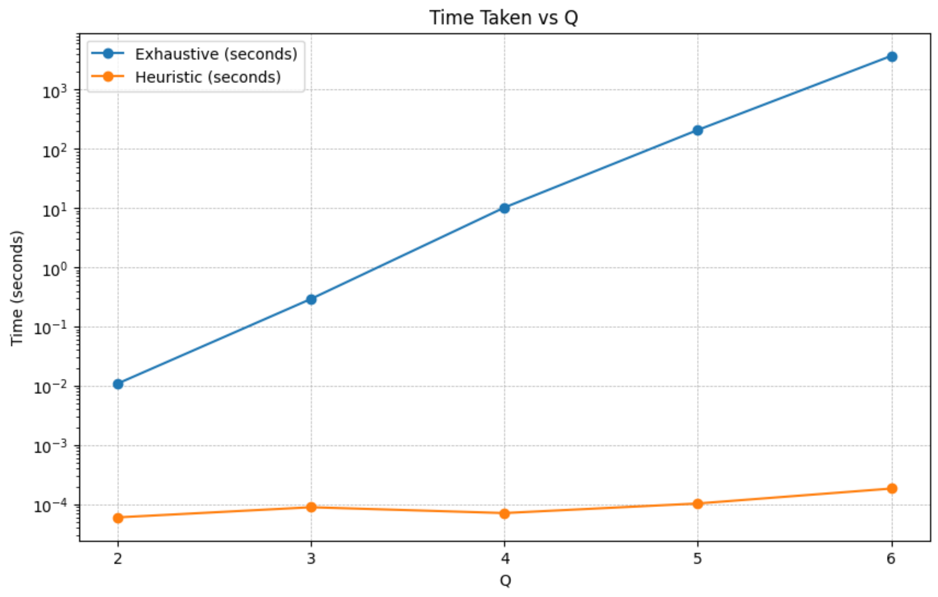


Figure 4.2 Time taken by exhaustive and heuristic methods as the number of Q increases.

Figure 4.2 illustrates the time taken by exhaustive and heuristic techniques as the number of Q increases in a pool of 50 devices. The exhaustive method shows exponential growth in execution time which is impractical for real-life scenarios. For instance, when $Q = 3$, the exhaustive method takes approximately 0.29 seconds, but as Q increases to 5, the time rises dramatically to 208.64 seconds, and at $Q = 6$, it reaches a very high value of 3710.64 seconds. This is due to the combinatorial nature in the number of possibilities the exhaustive method evaluates as Q increases. In contrast, the heuristic method demonstrates consistently low execution times, taking only 0.00006 seconds for $Q = 3$ and 0.00018 seconds for $Q = 6$.

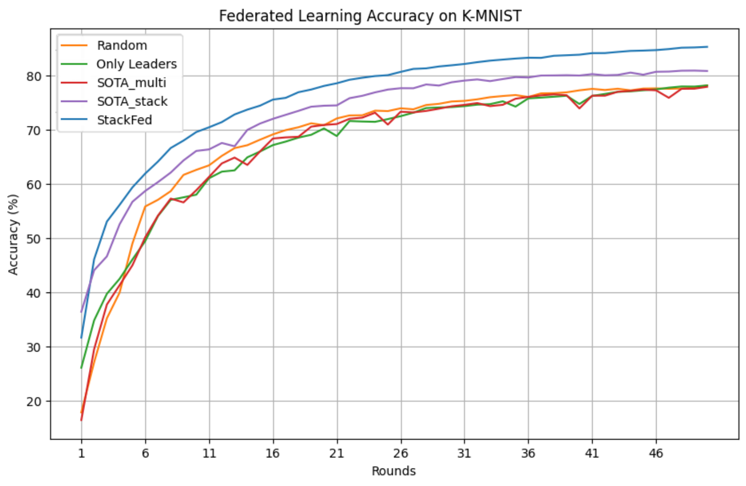
This efficiency arises from its ability to approximate solutions without evaluating all possible combinations, making it highly scalable and practical for larger values of Q.

4.3.2 Accuracy Convergence

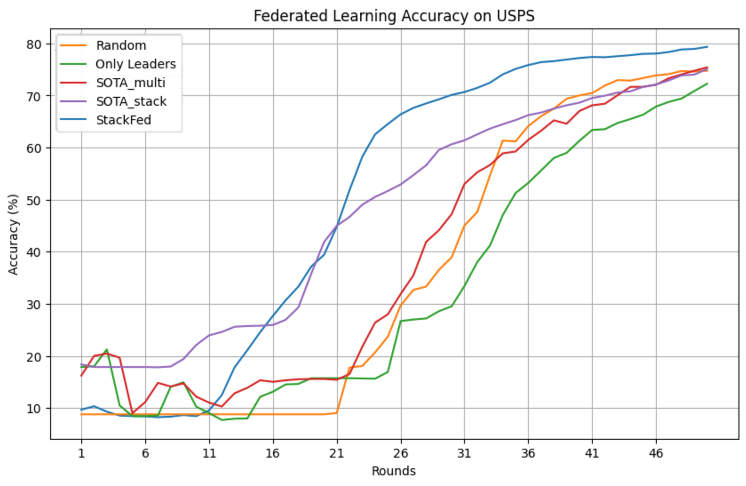
In this section, we examine the impact of our proposed framework on the global model's accuracy. The goal of this set of experiments is to compare the learning performance achieved through our client selection strategy against the other methods. These comparisons are conducted over three distinct datasets; FashionMNIST, K-MNIST, and USPS; each of which presents distinct distributional and representational challenges. We vary the coalition quota parameter $Q \in \{4, 5\}$ to explore how the number of followers per leader affects convergence behavior. The results of these experiments are plotted in Figures 4.3a, 4.3b, and 4.3c with having $Q = 4$, while we highlight the results with $Q = 5$ in Figures 4.4a, 4.4b, and 4.4c.

The accuracy results across the three datasets consistently demonstrate the superiority of the StackFed framework over all baselines and state-of-the-art approaches. On the K-MNIST dataset, StackFed achieves the fastest convergence and the highest final accuracy, surpassing all competitors by a margin of around 5%, largely due to early access to diverse and class-inclusive follower data. In the USPS dataset, where other methods struggle with slow learning and low early-round accuracy (e.g., Random and Only Leaders remain below 20% until round 20), StackFed accelerates past 70% by round 40, outperforming all others by up to 10%. With FashionMNIST dataset, StackFed maintains a lead in the final rounds, indicating its sustained generalization advantage. SOTA$_{stack}$ often performs second-best but shows slower improvement over time, while Random and Only Leaders plateau earlier, reinforcing the advantage of follower inclusion. These results clearly demonstrate that StackFed's coalition-based selection mechanism, significantly improves global model performance by leveraging data heterogeneity and inclusiveness across the client population.

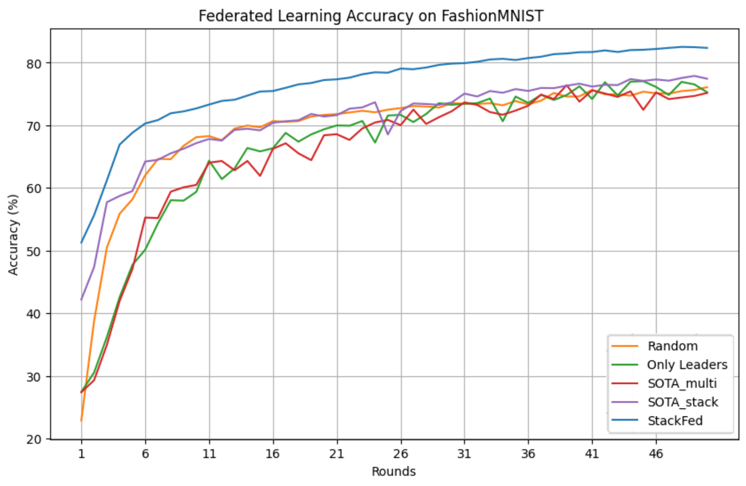
To further reinforce the advantages of the StackFed framework, we perform an extended set of experiments with $Q = 5$, and showcase the results in figure 4.4. Particularly, on the K-MNIST dataset, StackFed achieves a final accuracy exceeding 84%, maintaining a consistent lead of over the best-performing alternative (SOTA$_{stack}$), and showing noticeably faster convergence during the early and middle rounds. In the USPS dataset, StackFed demonstrates a sharp increase in accuracy starting from round 15, surpassing 75% by the end of the training. This is particularly important in a dataset characterized by significant class imbalance, where our coalition strategy effectively incorporates rare and diverse data contributions. In the FashionMNIST experiment, StackFed consistently outperforms all other methods, finishing with over 70% final accuracy, while SOTA$_{stack}$ and Random lag behind by around 5%. Across



(a) Accuracy on K-MNIST

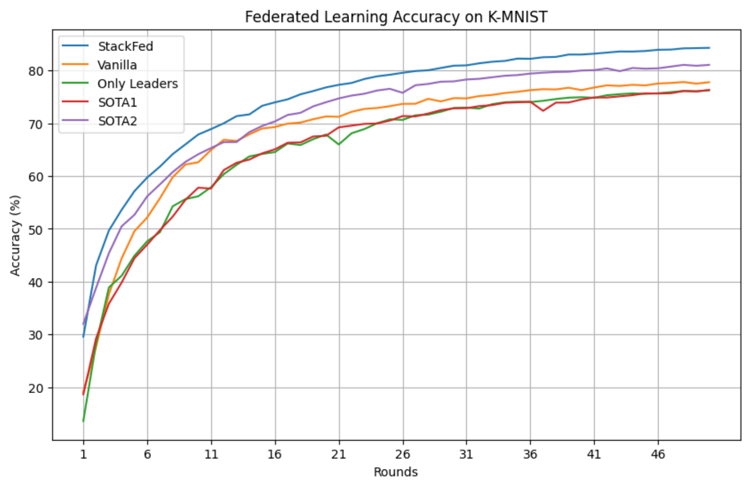


(b) Accuracy on USPS

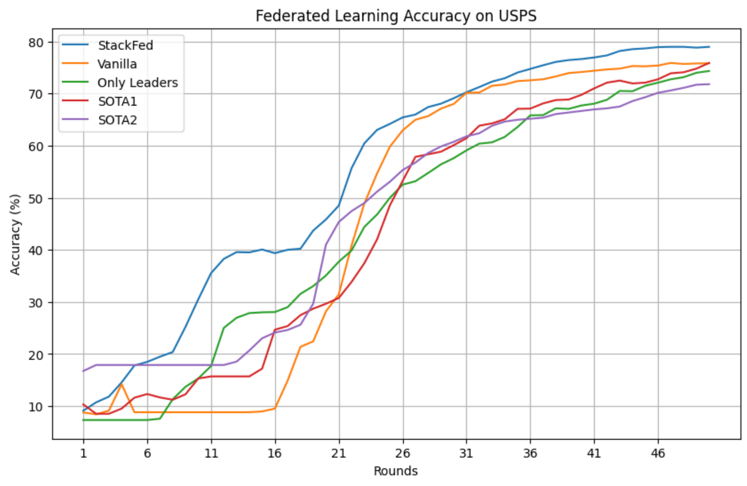


(c) Accuracy on FashionMNIST

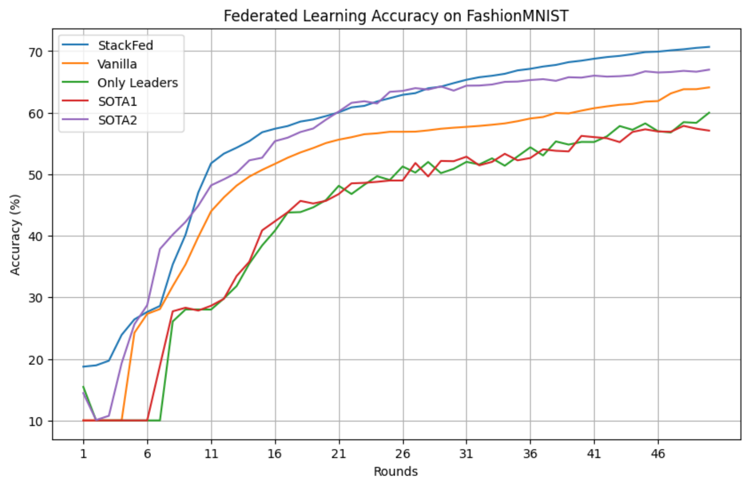
Figure 4.3 Global model accuracy over 50 communication rounds for different datasets using $Q = 4$.



(a) Accuracy on K-MNIST



(b) Accuracy on USPS



(c) Accuracy on FashionMNIST

Figure 4.4 Global model accuracy over 50 communication rounds for different datasets using $Q = 5$.

all datasets, increasing the coalition size to $Q = 5$ enables leaders to integrate a broader pool of follower clients, which enhances data diversity and accelerates convergence. These results affirm that StackFed's performance scales well with coalition size, offering both learning efficiency and improved model generalization across diverse federated environments.

4.3.3 Evaluating Class Imbalance and Data Inclusivity

Another major challenge in FL environments is the prevalence of data imbalance, where certain classes are overrepresented while others are scarcely present across the participating clients. This imbalance can significantly hinder model generalization, leading to biased predictions and lower accuracy, particularly on underrepresented classes. To evaluate this aspect, we introduce a *Class Diversity Index*, computed as the ratio between the minimum and maximum class representation (i.e., $\min \% / \max \%$) in the aggregated data of each approach. A higher value indicates better balance across all classes, as it reflects that no single class dominates or is neglected. By comparing this index across different baselines, we assess how well each method mitigates class imbalance and contributes to a fairer and more comprehensive learning process.

We run two set of experiments with $Q \in \{4, 5\}$. We highlight the results of the set of experiments ran with $Q = 4$ through figure 4.5, while we highlights the results of experiments with $Q = 5$ in figure 4.6.

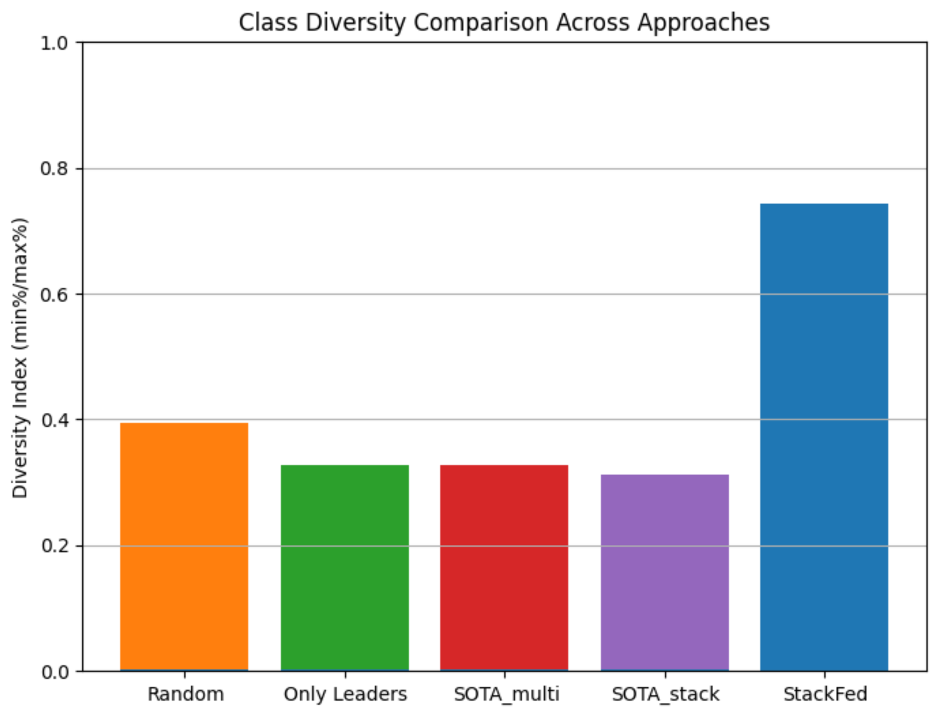


Figure 4.5 Class Diversity Index across selection strategies for $Q = 4$.

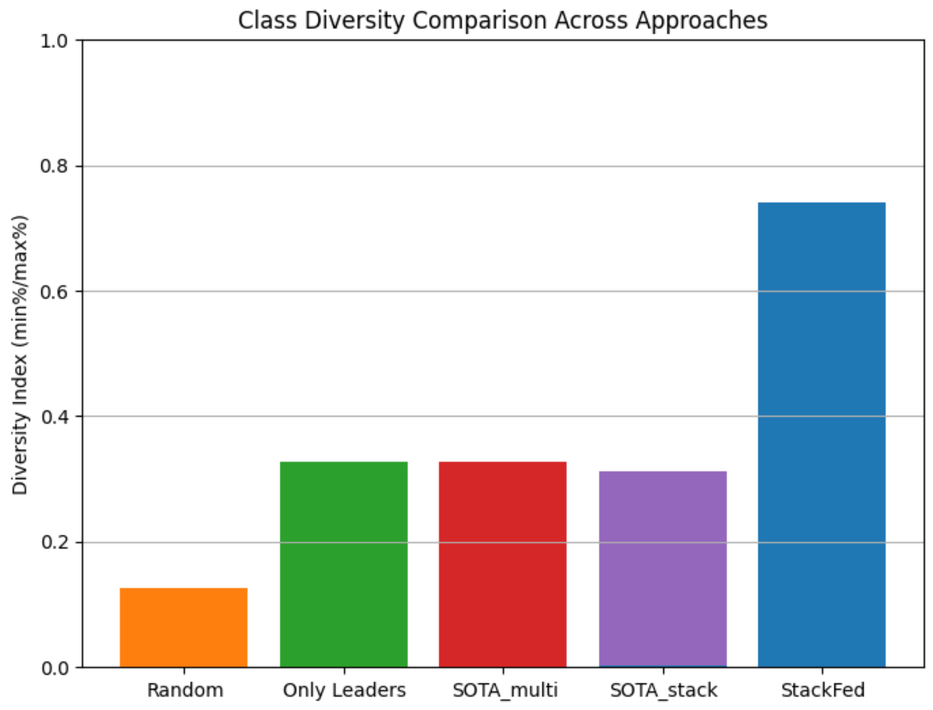


Figure 4.6 Class Diversity Index across selection strategies for $Q = 5$.

As shown in Figure 4.5, StackFed outperforms all baselines in terms of class diversity. While methods like Random, SOTA$_{multi}$, and SOTA$_{stack}$ exhibit diversity index values in the range of 0.31–0.39, StackFed achieves a much higher score of approximately 0.74. This indicates a far more balanced representation of all classes in the global dataset formed by the selected clients. The improvement is a direct result of StackFed's coalition mechanism, which allows leaders to partner with diverse followers, each contributing distinct and complementary class samples. In contrast, baselines that rely solely on resource-rich clients or random sampling tend to lack a deliberate mechanism to ensure class variety.

These findings confirm that StackFed addresses the critical issue of data imbalance, which is essential for training robust and unbiased models in real-world federated environments.

We extend the class diversity experiment using a larger coalition quota of $Q = 5$. The objective is to determine whether increasing the number of followers per leader allows for broader class inclusivity and better distribution balance in the aggregated data, which is essential for overcoming the data imbalance problem in federated learning. As in the previous setting, we compute the *Class Diversity Index* for each approach, defined as the ratio between the minimum and maximum class representation percentages.

The results presented in Figure 4.6 showcase that StackFed significantly improves data balance even under the larger $Q = 5$ setting. With a class diversity index of approximately 0.73, StackFed stands far above the baselines, where values range between 0.13 (Random) and 0.33 (SOTA$_{multi}$/Only Leaders). This highlights StackFed's consistent ability to include underrepresented classes in the training process, an effect amplified by the increased coalition size, which allows leaders to draw from a more diverse pool of follower clients. In contrast, Random and other baselines suffer from severe imbalance, often dominated by a few overrepresented classes due to random or resource-prioritized selection strategies.

4.3.4 Data Privacy

To address potential privacy and security concerns in our framework, we analyze the risk of metadata leakage during coalition formation and demonstrate how our approach implicitly mitigates it through entropy-aware client selection. In StackFed, followers share metadata such as data class distributions to be considered for selection. While this does not involve sharing raw data, it introduces a potential vector for inference attacks, particularly if a client's metadata is highly unique or skewed.

To quantify this, we compute the entropy of each client's class distribution. Entropy serves as a proxy for how balanced and diverse a client's data is; lower entropy implies skewed or narrow data, which may increase the risk of identification or profiling. We then compare the entropy values of all clients to those selected by StackFed.

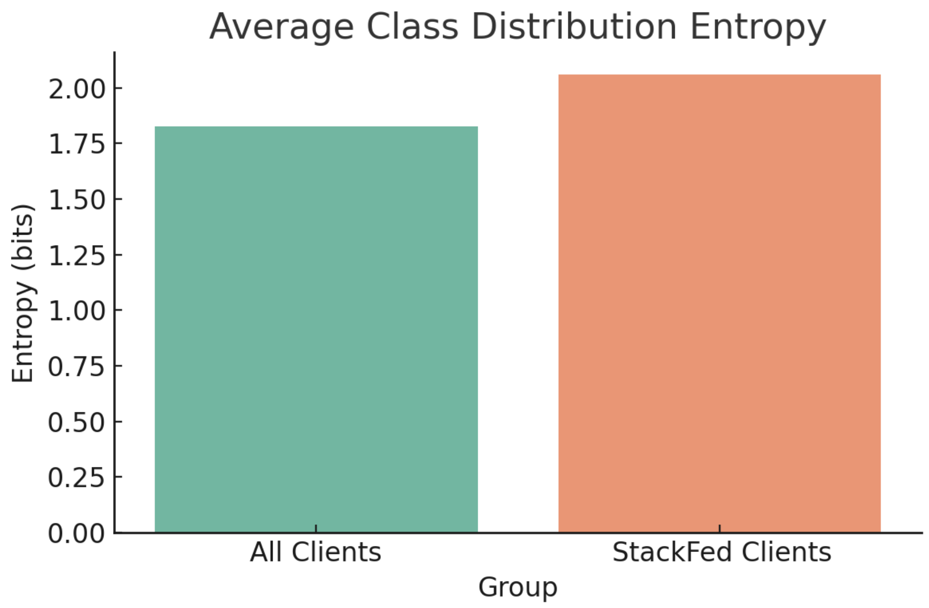


Figure 4.7 Average class distribution entropy of all clients compared to the clients selected by StackFed.

We analyze the class distribution diversity of selected clients by comparing the average entropy of all clients with that of the clients selected by StackFed. The results show that the top StackFed clients exhibit an average entropy of approximately 2.59 bits, compared to 2.13 bits for the general client population, an increase of around 21.6%. This indicates that StackFed consistently selects clients whose data is more balanced across multiple classes. Quantitatively, this improves the representativeness of the training data, leading to faster convergence and better model generalization. Furthermore, this benefits the including of clients with valuable data, and enhancing privacy by reducing the risk of inference or profiling attacks, as high-entropy metadata is less distinctive and harder to exploit. Such a behavior, reduces the risk of metadata-based inference, as it is harder to reverse-engineer the underlying data when class distributions are diverse. Hence, in addition to its performance and fairness benefits, StackFed implicitly promotes privacy-preserving client selection by favoring non-unique, generalizable data profiles.

4.4 Conclusion

In this work, we introduced StackFed, a novel FL framework that decentralizes client selection through a client-to-client Stackelberg game. By shifting leadership from the central server to resource-rich clients, StackFed enables coalition formation with resource-limited followers. We also proposed a data scoring mechanism that evaluates client contributions based on volume, class diversity, and distribution balance. This mechanism informs both coalition formation and utility optimization, which we formally model through a Stackelberg game, through algorithms for payment selection and optimization of leader utility. Our experiments show that StackFed consistently outperforms baselines and state-of-the-art approaches in accuracy, data diversity, and computational efficiency.

CHAPTER 5 ARTICLE 3: DESIGNING FEDERATED LEARNING MARKETPLACES: INCENTIVES, NETWORK DYNAMICS, AND DECENTRALIZED LEARNING

Submitted to IEEE Transactions on Network Science and Engineering on May 19, 2025.
Authors: Sarhad Arisdakessian, Osama Wehbi, Omar Abdul Wahab; Azzam Mourad, Hadi Otrok, Mohsen Guizani.

We continue our thesis and in this chapter focus on the challenge of finding effective ways to incentivize clients for continuous contributions. This aspect is crucial to maintaining the quality of the global model. Accordingly, we propose a novel FL marketplace comprising three main entities: the Network Owner (NO), the Task Owner (TO), and the clients. The network owner acts as the intermediary, managing the FL process, ensuring security, and facilitating interactions between the task owner and clients. Mainly, the network owner handles the traditional tasks of the federated server and more. Its services are enhanced beyond simple model aggregation. In addition to functioning as the federated server, the network owner assumes network management, client selection, reputation tracking, and incentive distribution, creating a structured and efficient ecosystem for both task owners and clients. The task owner is an entity that commissions the development of machine learning models, typically lacking the necessary resources but willing to pay to another entity to handle the task for them. On the other side, clients are data holders who contribute their datasets to the learning process in exchange for rewards. The main objective of our solution is to enable task owners to create models tailored to their needs while ensuring that clients are fairly compensated for their contributions.

Providing economic incentive [71] is an approach to motivate clients to participate in the FL processes and they align with the inherent context of decentralized systems that involve multiple clients with diverse data resources. Such mechanisms incentivize clients to actively participate while ensuring that the network owner can select the most valuable contributors without overpaying. This approach fosters a sustainable and economically viable ecosystem where economic incentives align with data quality and client participation. By integrating reverse auction games and internal reputation systems, it ensures fairness, strengthens client selection, enhances model accuracy and security, and guarantees a continuous supply of high-quality data, ultimately benefiting all stakeholders.

An examination of current literature and systems reveals interconnected challenges that collectively impede FL from evolving into a sustainable, inclusive, and economically viable ecosystem.

1. **Role separation and accountability**: Traditional FL architectures often conflate the roles of Task Owners and network operators, assuming a monolithic entity responsible for both initiating tasks and managing the FL process [104]. This lack of role differentiation can lead to conflicts of interest and reduced transparency. In this work, we promote a multi-stakeholder architecture that explicitly separates these roles, introducing a clear delineation of responsibilities between the task owner, network owner, and clients.

2. **Incentive mechanisms for client participation**: Traditional FL systems operate under the assumption that clients will participate altruistically or are sufficiently motivated by access to the global model. However, in practical scenarios, especially within cross-silo FL, clients often require direct, tangible incentives to contribute their data and computational resources [105]. Our framework addresses this by implementing a reverse auction mechanism, allowing clients to bid their participation price, thereby aligning incentives with individual client motivations.

3. **Reputation systems, behavioral enforcement & fairness**: Ensuring trustworthy behavior from all participants in FL is crucial for the integrity of the learning process [106]. While some systems have introduced reputation mechanisms to evaluate client reliability, these often lack persistence and punitive measures for dishonest behavior [107]. For example, reputation scores are often resettable, non-binding, or subjective. Our proposed reputation system builds upon these ideas by maintaining persistent reputations and enforcing penalties for malicious actions, thereby fostering a more secure and accountable FL environment.

5.1 Role-Based Formalization of the FL Marketplace Network

To enable formal analysis and mechanism design, we model the FL marketplace as a structured interaction between distinct economic agents. We describe the participating entities and define their operational roles mathematically as follows.

$$\mathcal{F} = (\mathcal{T}, NO, \mathcal{I}, \mathcal{B}, \mathcal{R}, \mathcal{A}, \mathcal{S}), \tag{5.1}$$

Where

- $\mathcal{T}$: Set of Task Owners (TOs), each defining learning tasks
- NO: A Network Owner managing orchestration

- $\mathcal{I}$: Set of Clients providing data and model updates
- $\mathcal{B}$: Set of client bids submitted for each task
- $\mathcal{R}$: Mapping of reputation scores
- $\mathcal{A}$: Auction mechanism governing client selection
- $\mathcal{S}$: Server-side orchestration and aggregation logic

5.1.1 Task Owner:

The task owner wants to train a specific model or improve an existing one but lacks the data and the system resources. However, it will pay a third-party provider (i.e., the network owner) to handle this task. Task owners represent the *demand* side of the marketplace, seeking data and computational resources to achieve their objectives.

Each task $T_t \in \mathcal{T}$ is defined as

$$T_t = (\mathcal{L}_t, \Theta_t, B_t), \tag{5.2}$$

Where

- $\mathcal{L}_t$: Learning objective (e.g., classification task)
- Θ_t: Model and data requirements
- B_t: Budget allocated for the task

5.1.2 Network Owner:

The network owner acts as the FL service provider and manages the entire process. Its main task is to function as an intermediary between task owners and clients. The network owner is responsible for establishing the system, managing the backend servers, and streamlining the process for both task owners and clients. A network owner acts as an intermediary via the mapping

$$\Phi_n : (\mathcal{T}, \mathcal{I}) \to \mathcal{M}_t. \tag{5.3}$$

This function maps each task and client set to a trained global model $\mathcal{M}_t$, managing selection, incentives, and aggregation.

5.1.3 Clients:

Clients represent the *supply* side of the marketplace. They are data owners; such as individuals, edge devices, or organizations; that possess relevant datasets and computational resources. By participating in the FL process, clients contribute to training the model while receiving compensation in return. Their involvement is economically motivated, with decisions influenced by factors such as cost of participation, reputation, and expected rewards. Each client $i \in \mathcal{I}$ is defined as

$$\theta_i = (\mathcal{D}_i, \kappa_i, R_i). \tag{5.4}$$

Where

- $\mathcal{D}_i$: Local dataset held by client i
- κ_i: True participation cost
- R_i: Reputation score of the client

The formulation of the marketplace encapsulates the economic and computational primitives of the FL marketplace, treating task owners, network owners, and clients as strategic agents linked through bids, reputation dynamics, and an incentive-compatible auction. This role-based model clarifies information flow and accountability, while laying the mathematical groundwork for the utility analysis presented in the subsequent sections in this chapter.

The separation of roles into task owner, network owner, and clients is a core design choice in the proposed marketplace, enhancing accountability and transparency over monolithic FL architectures. By explicitly structuring interactions among roles, this formulation also facilitates the resolution of potential conflicts; if they arise; through well-defined conflict resolution mechanisms. The current implementation incorporates mechanisms to address these issues and is architected to support additional governance layers in case of need, in deployments that require stronger guarantees.

From an economic perspective, it is inherently against the NO's long-term interest to cheat or mistreat clients. As a competitive marketplace, the NO's ability to attract TOs depends on maintaining a sufficiently large and diverse client base to meet task requirements. Mistreatment or biased selection risks discouraging clients from participating, pushing them toward competing networks. This erosion of the client pool would directly impair the NO's capacity to advertise tasks, recruit high-quality participants, and deliver reliable training services to TOs. In effect, the marketplace's competitive dynamics align the NO's incentives with fair and transparent treatment of clients, creating an economic self-regulation mechanism in addition to formal governance.

On other hand, conflicts may arise between the NO and clients, particularly when participants contest how their contributions are valued or how their reputations are adjusted. For example, a client may dispute being excluded from a training phase despite submitting a competitive bid, or argue that a sharp drop in their reputation score was unjustified and disproportionately punitive. Given that reputation directly impacts future selection opportunities, a negative adjustment, even if accidental, can have substantial economic consequences for the affected client.

The marketplace's design addresses these risks at two levels. At its core, the architecture enforces transparent, deterministic scoring rules: the formulas for the Data Contribution Value Function (DCVF), contribution score (ϕ_i), and reputation update mechanism are publicly defined within the system's operational logic. This allows any participant to independently verify the outcome of their participation. Furthermore, the system can support an appeal channel through which clients can submit evidence, such as logs of model updates or proof of data coverage, to request a review of disputed valuations or penalties. All selection and reputation decisions are logged, providing a verifiable history that can be used for dispute resolution and post-hoc audits.

5.2 Reputation System

In this section, we detail the design and mathematical formulation of the reputation system embedded within our FL marketplace. The reputation mechanism plays a central role in maintaining long-term behavioral integrity, deterring strategic dishonesty, and promoting fairness across participants.

5.2.1 Reputation System Characteristics

The proposed reputation framework is governed by four foundational principles:

1. **Proprietary**: Each marketplace network maintains its own self-contained reputation environment. Reputation scores are not portable across networks unless a cross-validation agreement is explicitly established.

2. **Internal**: Reputation is computed entirely within the network, using verifiable participation data and behavior logs. No external ratings, imported history, or off-chain

verification mechanisms are used.

3. **Perpetual**: Reputation accumulates over time and cannot be reset arbitrarily. It is subject only to behavioral updates or inactivity decay, allowing clients to build or lose trust through consistent engagement.

4. **Punitive**: Dishonest behavior is penalized more harshly than honest behavior is rewarded, i.e., $n \gg m$, to discourage rational misbehavior and promote cooperation over opportunism.

5.2.2 Initial Reputation

New clients entering the system are initialized with a neutral reputation value:

$$R_i^0 = 0. \tag{5.5}$$

This decision reflects the proprietary and internal nature of the system, where trust must be built organically within each marketplace network. Clients with outstanding history in other systems must re-establish their value locally. The initial neutrality ensures a level playing field and allows newcomers to prove their reliability through incremental participation. This design also creates a defense against Sybil attacks, wherein adversaries generate multiple fresh identities to manipulate the auction, disrupt the training process, or poison the global model. By requiring every new identity to build reputation through verifiable and sustained honest participation, the system ensures that trust is earned rather than given, making large-scale Sybil attacks significantly more costly and time-consuming.

Positive Welcome Prior

While the zero-start policy offers strong protection against Sybil attacks by requiring all clients to build trust from scratch, it can be further enhanced to encourage faster integration of valuable newcomers. To this end, we introduce the optional *Positive Welcome Prior*, an initialization scheme in which new clients receive a small positive base reputation at registration. This approach preserves the security benefits of the zero-start framework while enabling legitimate participants with high-quality and diverse datasets to enter the marketplace competitively from the outset.

$$R_i^0 = 0 + R_{base}, \quad where \quad 0 < R_{base} < R_{\min}. \tag{5.6}$$

Here, R_{base} is chosen to be large enough to allow competitive bidding for newcomers with strong data contributions, yet small enough to prevent automatic preference over well-established, trustworthy clients. We empirically show that even a modest Positive Welcome Prior alters client selection and improves model accuracy (Section 5.5.5).

Furthermore, broader and optional design alternatives to address the limitations of the zero-start policy are discussed hereafter: In its early deployment, the marketplace faces a cold-start challenge, where neither task owners, nor clients possess established reputations, and task volume may be too low to sustain engagement. A structured seeding strategy addressing both supply and demand can mitigate this.

Demand-side (TO recruitment): The NO can introduce seed tasks; low-risk, internally funded or partner-supported training jobs with measurable outcomes (e.g., benchmark model gains) to demonstrate platform value. Early TOs may receive temporary fee waivers, reduced commissions, or guaranteed SLAs to lower adoption risk.

Supply-side (Client recruitment): The NO can offer short-term participation subsidies, guaranteeing a minimum payment alongside performance-based rewards in early phases. Founder bonuses in the PIPP reputation system can grant early clients a future competitive advantage, contingent on sustained honest participation. Before onboarding external TOs, the NO can act as its own initial TO. Network effects can be accelerated via invite-and-reward programs, granting bonuses for onboarding new verified participants. This phased approach establishes a virtuous cycle: seed tasks attract and retain clients, whose presence makes the marketplace more appealing to TOs, driving self-sustaining growth.

5.2.3 Reputation Updates

The reputation update process is decoupled from the bidding mechanism. In our design, clients submit a one-time bid for a task at its initiation. If selected, they participate in the entire learning process across multiple rounds. There is no repeated bidding per round. This avoids excessive overhead and latency. (We note that the decoupling in this context refers to the reputation updates that take place through different times, even though reputation values influence future selections.)

Once participation begins, the client's reputation is updated at the end of each round based on the behavioral classification of their contribution and honesty. Let $\mathcal{H}_i^{round} \in \{honest, dishonest\}$ denote the behavior of client i at a given round. The reputation update rule is given by

$$R_i^{round+1} = \begin{cases} R_i^{round} + m & if \mathcal{H}_i^{round} = honest \\ R_i^{round} - n & if \mathcal{H}_i^{round} = dishonest, \end{cases} \tag{5.7}$$

Where

- $m > 0$ is the fixed positive reward for compliant behavior,
- $n \gg m$ is the penalty for malicious or free-riding participation,
- R_i^{round} is the client's reputation at given round.

This formulation ensures honest clients accumulate score gradually, while dishonest clients are severely penalized and must compensate via multiple honest rounds.

Behavior classification $\mathcal{H}_i$ is determined via internal auditing by the network owner. The network uses $\mathcal{V}(\mathcal{M}_i)$, a behavior verification function applied to the local model update $\mathcal{M}_i$ from client i in a given round, to classify participation as honest or dishonest:

- **Free-riding detection:** Similarity-based checks (e.g., FoolsGold [108], cosine similarity, Euclidean distance) are used to detect unmodified or plagiarized updates.
- **Contribution testing:** A validation dataset (if available) $\mathcal{D}_{val}$ is used to evaluate whether $\mathcal{M}_i$ produces a suspicious outputs.
- **Inactivity recognition:** Clients that fail to upload model updates are flagged as inactive. However, the reputation mechanism distinguishes between strategic non-participation and involuntary dropouts. Non-malicious upload failures are treated as inactivity, subjecting the client to natural reputation decay rather than the immediate, punitive penalties ($n \gg m$) reserved for verifiable malicious behavior. This approach accounts for the stochastic nature of wireless connectivity, ensuring that honest but resource-constrained devices are not unfairly marginalized.

This evaluation ensures that only actively contributing, honest clients are rewarded, while dishonest behavior leads to penalization under Eq. 5.7.

5.2.4 Reputation Decay

To promote sustained engagement and discourage prolonged inactivity, the marketplace applies a time-based decay that reduces a client's reputation when they have not bid or participated for an extended period. For each client i, the system stores the last update timestamp

t_i. Decay is applied only when the reputation R_i is accessed:

$$R_i(t) \;=\; \max\Big(0,\, R_i(t_i)\, e^{-\delta\,(t-t_i)}\Big),$$

where $\delta > 0$ is the continuous-time decay rate and t is the current timestamp. After decay, the behavioral update is applied ($+m$ for honest, $-n$ for dishonest participation), and the timestamp is set to $t_i \leftarrow t$. This event-driven (i.e., lazy) evaluation ensures computational efficiency, since decay is computed only for the actively evaluated set $|S_r|$ rather than the entire pool n, achieving $\mathcal{O}(|S_r|)$ complexity in practice with $|S_r| \ll n$.

5.2.5 Clients' Comprehensive Reputation Formula

The complete evolution of a client's reputation combines event-driven behavioral updates and continuous-time decay due to prolonged marketplace inactivity. We define the unified update formula as

$$R_i(\theta+\tau) = \begin{cases} R_i(\theta) + m & if \mathcal{H}_i(\theta) = \text{honest and active} \\ \max(0,\, R_i(\theta) - n) & if \mathcal{H}_i(\theta) = \text{dishonest and active} \\ \max\big(0,\, R_i(t_i)\, e^{-\delta\,(t-t_i)}\big), & if \text{ inactive for } \tau \text{ time units,} \end{cases} \tag{5.8}$$

Where:

- $R_i(\theta)$: Reputation score of client i at time θ,
- m: Positive increment for honest participation,
- n: Penalty for dishonest behavior,
- $\delta > 0$: Continuous-time exponential decay rate,
- τ: Duration of inactivity,
- $\mathcal{H}_i(\theta)$: Behavioral classifier.

This formula constitutes our **PIPP** reputation model: *Proprietary, Internal, Perpetual, and Punitive.* It ensures that reputation evolves predictably; rewarding honest participation while discouraging marketplace-wide inactivity. The exponential decay mechanism reflects the continuous erosion of trust in idle clients, motivating regular engagement through bidding or task participation.

5.3 Data Contribution Value Function (DCVF)

The *Data Contribution Value Function* $V(\mathcal{D}, Q)$ serves as a critical metric for evaluating each client's contribution within the FL marketplace. It quantifies the relevance and sufficiency of a client's data in meeting the NOs requirements, capturing both class coverage and data volume. By integrating this value function with reputation scores and bid factors, the selection process becomes more structured and data-driven, enabling the NO to optimize utility by prioritizing high-quality contributions for the task at hand.

To better handle scenarios where certain classes are more critical to the task than others, we introduce a weighted formulation of the DCVF. This allows the NO to assign explicit importance scores $w_d > 0$ to each required class $d \in \mathcal{D}$, reflecting domain priorities or the scarcity of certain data types. This weighted version allows the system to handle imbalanced and non-IID data distributions more effectively by assigning higher importance to rare or underrepresented classes. The formal representation of the weighted Data Contribution Value Function is:

$$V_i(\mathcal{D}_i, Q) = \gamma \cdot C_i^{(w)}(\mathcal{D}_i) \times \rho \cdot S_i^{(w)}(Q), \tag{5.9}$$

where:

- $V_i(\mathcal{D}_i, Q)$ is the weighted contribution value of client i's data,
- $\gamma > 0$ is a weight factor for class coverage,
- $\rho > 0$ is a weight factor for data quantity,
- $C_i^{(w)}(\mathcal{D}_i)$ is the weighted class coverage function,
- $S_i^{(w)}(Q)$ is the weighted quantity sufficiency function.

The weighted class coverage is given by:

$$C_i^{(w)}(\mathcal{D}_i) = \frac{\sum_{d \in \mathcal{D}_i \cap \mathcal{D}} w_d}{\sum_{d \in \mathcal{D}} w_d}, \tag{5.10}$$

where:

- $\mathcal{D}$ is the set of required classes for the task,
- $\mathcal{D}_i$ is the set of classes available in client i's dataset,

- w_d is the importance weight for class d.

The weighted quantity sufficiency is given by:

$$S_i^{(w)}(Q) = \frac{\sum_{d \in \mathcal{D}} w_d \cdot \min\left(1, \frac{q_{i,d}}{r_d}\right)}{\sum_{d \in \mathcal{D}} w_d}, \tag{5.11}$$

where:

- $q_{i,d}$ is the quantity of data instances for class d provided by client i,
- r_d is the required quantity for class d as defined by the task owner.

The $\min(1, q_{i,d}/r_d)$ term ensures that supplying more than the required amount for a class does not inflate the sufficiency score, preventing oversupply in one class from compensating for shortages in others. The multiplicative design in Eq. 5.9 enforces that both diversity and sufficiency contribute jointly to the overall score. This prevents gaming strategies and ensures that the selected clients collectively provide a balanced, high-utility dataset that meets both the diversity and volume requirements of the task. This weighted formulation generalizes DCVF by allowing the network owner to incorporate domain-specific priorities into the scoring process. In scenarios where all classes are equally important, setting $w_d = 1$ for all d recovers the original unweighted formulation.

5.4 Reverse Auction Games

Reverse auction games constitute a sub-field in the broader field of *mechanism design* [109]. This domain focuses on designing rules or mechanisms to achieve a specific objective, often under conditions of private information. In particular, reverse auction games differ from traditional (forward) auction games in several key ways:

1. In reverse auction games, the roles of the buyer and seller are reversed. Unlike traditional auctions where buyers compete to pay the highest price to a seller, reverse auctions involve a single buyer receiving bids from multiple sellers, who compete to offer the lowest price.
2. Traditional auctions often have ascending bids where the highest bidder wins, whereas reverse auctions effectively have descending competition, where lower bids are more competitive.

3. In traditional auctions, multiple buyers compete to purchase from a single seller. In reverse auctions, multiple sellers compete to sell to a single buyer.

4. In traditional auctions, the objective is to maximize the sale price and benefit the seller. In reverse auctions, the objective is to minimize the purchase price while satisfying quality or coverage requirements for the buyer.

In our setting, the network owner plays the role of the buyer and seeks cost-effective, high-quality training contributions from a pool of client devices, which act as sellers.

5.4.1 Single-Shot Reverse Auction in Our Marketplace

Our proposed system employs a stateless and synchronous single-shot reverse auction model in which all eligible clients submit their bids once at the start of the task lifecycle ($t = 0$). The NO collects these bids within a predefined window and executes the selection algorithm immediately upon closure, ensuring fairness by evaluating all participants under identical conditions and avoiding latency-induced discrepancies.

The selected set of clients $\mathcal{I}^\star$ remains fixed for all $Rounds_{\text{total}}$ training rounds of the task, eliminating the need for repeated re-selection or re-bidding. This reduces computational overhead, avoids strategic fatigue, and preserves cohort stability. If a client drops out mid-task, the round proceeds with the remaining cohort, and the absence is recorded by the PIPP reputation system as a missed contribution, triggering proportional penalties that lower future selection probability.

To maintain continuity without introducing disruptive re-selection, replacements can be drawn from a pre-ranked standby list generated during the initial auction. Persistent reputation tracking ensures returning clients retain their accumulated scores, deterring churn for reputation resets, while new entrants begin at a neutral baseline and must build trust over time.

5.4.2 Principal and Strategic Agents

We model the reverse auction-based FL marketplace as a mechanism design setting consisting of a single principal and a set of strategic agents:

- **Strategic agents (clients).** Let $\mathcal{I} = \{1, 2, \dots, n\}$ denote the set of clients participating as data sellers. Each client $i \in \mathcal{I}$ possesses a private type (κ_i, D_i), where κ_i is the true participation cost and D_i is the local dataset held privately by the client.

The NO maintains, for each client, a reputation score R_i (via PIPP) and computes a data valuation $V_i(D_i, Q)$ based on the task requirements. In the stage game, each client strategically chooses a bid $\hat{P}_i \in \mathbb{R}_{\geq 0}$ indicating the requested compensation.

- **Mechanism designer (Network Owner).** The NO defines an allocation and payment mechanism X that maps the bid profile $\hat{\boldsymbol{P}} = (\hat{P}_i)_{i \in \mathcal{I}}$ to (i) a selected subset of clients $\mathcal{I}^\star \subseteq \mathcal{I}$ and (ii) payments $\boldsymbol{p} = (p_i)_{i \in \mathcal{I}}$. The mechanism aims to maximize the NO's utility (defined below) subject to budget and data-coverage constraints.

Hereafter, we formalize the reverse auction as a single-parameter mechanism and analyze its incentive properties.

5.4.3 Mechanism-Design Formulation

We formulate the reverse auction as a single-parameter mechanism-design problem in which the NO acts as the buyer and the clients act as strategic sellers. Each client $i \in \mathcal{I}$ is characterized by a private type

$$\theta_i = (D_i, \kappa_i, R_i).$$

where D_i is the local dataset, κ_i is the unobserved true participation cost, and R_i is the reputation score maintained by the Network Owner.

Each client strategically selects an action

$$\hat{P}_i \in A_i := \mathbb{R}_{\geq 0},$$

representing the bid submitted to the reverse auction.

The stage game is described by the mechanism

$$\mathcal{G} = \Big(\mathcal{I}, \{A_i\}_{i \in \mathcal{I}}, \{U_i\}_{i \in \mathcal{I}}, X, U_{\text{NO}}\Big),$$

where:

- $\mathcal{I}$ is the set of clients,
- $A_i = \{\hat{P}_i \in \mathbb{R}_{\geq 0}\}$ is the action space,
- U_i is the client's utility,
- X maps bids to selection indicators,

- U_{NO} is the NO's utility under X.

We note that the reputation R_i and the data valuation $V_i(D_i, Q)$ are not part of the clients' strategy space. R_i is updated by the network owner from observed behavior during training, which also computes $V_i(D_i, Q)$ from the data statistics and task requirements. Both are therefore treated as exogenous parameters in the stage game, and strategic behavior is limited to price reporting.

5.4.4 Payment Rule, Critical Bid, and Client Utility

Each client strategically reports a bid $\hat{P}_i \in \mathbb{R}_{\geq 0}$ representing the compensation they request for participating in the learning task. For fixed reports from the other clients $\hat{\boldsymbol{P}}_{-i}$ and for fixed data valuation $V_i(D_i, Q)$ and reputation R_i, the allocation rule determines whether client i is selected.

Because the scoring function

$$\varphi_i(\hat{P}_i) = V_i(D_i, Q) + \alpha R_i - \beta \hat{P}_i, \tag{5.12}$$

is strictly decreasing in $\hat{P}_i$, the allocation rule is assumed to be monotone in the bid, which is a standard requirement for truthful single-parameter mechanisms.

Assumption 1 (Bid-monotone allocation)**.** *If $x_i(\hat{P}_i, \hat{\boldsymbol{P}}_{-i}) = 1$ for some $\hat{P}_i$, then $x_i(\hat{P}'_i, \hat{\boldsymbol{P}}_{-i}) = 1$ for all $\hat{P}'_i < \hat{P}_i$.*

Assumption 1 asserts that lowering a bid cannot hurt a client's allocation when other bids are fixed. Together with critical-value payments, this standard condition yields dominant-strategy incentive compatible (DSIC) in single-parameter procurement. We adopt bid-monotonicity as a sufficient design condition for our allocation rule under feasibility constraints; settlement payments are then computed via the induced critical values.

Critical Bid. For client i, the *critical bid* is the highest bid the client could have submitted while still remaining selected under the allocation rule:

$$\hat{P}_i^{\text{crit}}(\hat{\boldsymbol{P}}_{-i}) = \sup\left\{p \geq 0 \;\middle|\; x_i\Big(p, \hat{\boldsymbol{P}}_{-i}\Big) = 1\right\}, \tag{5.13}$$

where $x_i(\cdot) \in \{0, 1\}$ denotes the allocation rule. Intuitively, the critical bid is the threshold beyond which increasing the bid causes client i to lose its place to another competitor.

Payment Rule. Under Assumption 1, a standard characterization of truthful single-parameter procurement mechanisms implies that DSIC can be implemented via critical-value payments [110]. Accordingly, each selected client is paid its critical bid:

$$p_i(\hat{\boldsymbol{P}}) = \begin{cases} \hat{P}_i^{\text{crit}}(\hat{\boldsymbol{P}}_{-i}), & \text{if } x_i(\hat{\boldsymbol{P}}) = 1, \\ 0, & \text{otherwise.} \end{cases} \tag{5.14}$$

Thus, selected clients receive the maximum amount they could have bid and still remained selected. This ensures that clients who bid truthfully cannot improve their utility by misreporting their true cost.

Client Utility. Each client incurs a private (unobservable) cost κ_i for participating. The utility is therefore

$$U_i(\hat{P}_i, \hat{\boldsymbol{P}}_{-i}) = \begin{cases} \hat{P}_i^{\text{crit}}(\hat{\boldsymbol{P}}_{-i}) - \kappa_i, & \text{if } x_i(\hat{\boldsymbol{P}}) = 1, \\ 0, & \text{otherwise.} \end{cases} \tag{5.15}$$

Because the allocation is monotone in $\hat{P}_i$, there exists a threshold $\hat{P}_i^{\text{crit}}(\hat{\boldsymbol{P}}_{-i})$ such that client i is selected iff $\hat{P}_i \leq \hat{P}_i^{\text{crit}}$. Moreover, under the critical-value rule, the payment equals this threshold and is therefore independent of the exact bid as long as $\hat{P}_i$ remains below the threshold. Hence, any deviation that keeps $\hat{P}_i \leq \hat{P}_i^{\text{crit}}$ yields the same payment, while any deviation with $\hat{P}_i > \hat{P}_i^{\text{crit}}$ yields zero utility due to non-selection.

5.4.5 Incentive Compatibility and Equilibrium

A mechanism is dominant-strategy incentive compatible (DSIC) if truthful reporting of the private type is a weakly dominant strategy, i.e., truthful bidding never yields lower utility than any deviation, regardless of other agents' actions. In our setting, this means that reporting the true cost κ_i is weakly better than any other bid $\hat{P}_i'$ for all clients i.

Proposition 1 (Truthfulness). *Under bid-monotonicity and the critical-value payment rule, truthful bidding is a weakly dominant strategy:*

$$U_i(\kappa_i, \hat{\boldsymbol{P}}_{-i}) \geq U_i(\hat{P}_i', \hat{\boldsymbol{P}}_{-i}), \qquad \forall \hat{P}_i' \in A_i.$$

Proof. The proof follows the standard characterization for single-parameter procurement auctions [110]. Bid-monotonicity implies that there exists a critical value $\hat{P}_i^{\text{crit}}(\hat{\boldsymbol{P}}_{-i})$ such that client i is selected if and only if $\hat{P}_i \leq \hat{P}_i^{\text{crit}}$. The critical-value payment rule makes p_i independent of the specific bid as long as $\hat{P}_i$ remains below this threshold, while bidding above the threshold yields zero utility. Because the payment p_i equals the critical value $\hat{P}_i^{\text{crit}}$, it is independent of the particular bid as long as $\hat{P}_i \leq \hat{P}_i^{\text{crit}}$. Thus, any misreport below the critical value yields the same payment and hence the same utility, while any bid above the threshold yields utility 0.

Equilibrium Interpretation. As truthful bidding weakly dominates all deviations, and the profile $\hat{P}_i = \kappa_i$ for all i constitutes a dominant-strategy equilibrium of the stage game, and hence also a Nash equilibrium. Given our approach focuses on the one-shot auction, the dominant-strategy property ensures that rational clients have no incentive to deviate from truthful cost reporting in any single auction instance. Since every player has a weakly dominant strategy, the truthful profile forms a dominant-strategy equilibrium, which is a Nash equilibrium.

5.4.6 Network Owner Utility and Constraints

The utility function for the NO, denoted as U_{NO}, captures the overall benefit derived from managing the FL process. It is defined as

$$U_{NO} = \sum_{i \in \mathcal{I}^\star} V_i(D_i, Q) \; + \; \alpha \sum_{i \in \mathcal{I}^\star} R_i \; - \; \beta \sum_{i \in \mathcal{I}^\star} p_i, \tag{5.16}$$

where

- $\mathcal{I}^\star$ is the set of selected clients,
- $V_i(D_i, Q)$ is the data contribution value of client i,
- R_i is the reputation score of client i,
- p_i is the payment made to client i,
- α and β are weights assigned to reputation and payment, respectively.

The NO seeks to maximize U_{NO} by selecting a cohort $\mathcal{I}^\star$ that (i) satisfies the task's per-class data-quantity requirements and (ii) respects an operational planning budget during client selection. These feasibility requirements are enforced by the client selection algorithm (Algorithm 9), which constructs $\mathcal{I}^\star$ using the reported bids and coverage statistics. After selection, settlement payments are computed using critical values to ensure dominant-strategy incentive compatibility. The resulting settlement expenditure, which may exceed the planning budget due to critical-value payments, is quantified and the selected clients are paid accordingly.

5.4.7 Heuristic Client Selection Algorithm

Given that the marketplace can potentially involve a large number of clients in real-world scenarios, especially in cross-device FL settings, ensuring scalability is crucial. Evaluating all possible client combinations to select the best set of participants has a complexity of $\mathcal{O}(2^n)$, which is computationally prohibitive.

To address this challenge and make the system applicable in practice, we adopt a heuristic method and develop a greedy algorithm (Alg. 9) to perform client selection with tractable complexity. We define the individual contribution score of client i, denoted by φ_i, as

$$\varphi_i = V_i(D_i, Q) + \alpha R_i - \beta \hat{P}_i, \tag{5.17}$$

where

- φ_i: contribution score of client i,
- $V_i(D_i, Q)$: data contribution value,
- R_i: reputation score of client i,
- $\hat{P}_i$: bid price submitted by client i,
- α, β: weight factors controlling the influence of reputation and price, respectively.

This score combines the client's data value, reputation, and bid price, and serves as the selection criterion in our heuristic algorithm.

Algorithm 9 selects the most suitable clients to fulfill the NO's data class and quantity requirements. It ranks all candidates based on their contribution score φ_i, which combines data value, reputation, and bid price. The algorithm accepts three inputs: (1) a list of potential clients, (2) required data classes, (3) minimum quantity thresholds, and (4) Planning budget. It initializes data structures to track selected clients, covered classes, and current class quantities. Each client's contribution is computed and stored as a tuple, and the list is sorted in descending order of φ_i. The algorithm then iteratively selects clients who add new data classes or insufficiently covered quantities, updating the coverage state after each selection. It terminates early once all class and quantity requirements are satisfied.

Algorithm 9 Heuristic Client Selection Algorithm

```
Require: Candidate clients $\mathcal{I}$; required classes $\mathcal{D}$; per-class minimums $\{r_d\}_{d\in\mathcal{D}}$;
Require: planning budget $B_{NO}^{\mathrm{sel}}$
Ensure: Selected clients $\mathcal{I}^\star$

1: $\mathcal{I}^\star \leftarrow [\,]$, $covered\_classes \leftarrow \emptyset$, $current\_qty[d] \leftarrow 0 \quad \forall\, d \in \mathcal{D}$
2: $ranked\_clients \leftarrow [\,]$, $total\_bid \leftarrow 0$
3: for all $i \in \mathcal{I}$ do
4:     $\varphi_i \leftarrow$ ContributionScore($i$)
5:     append $(\varphi_i, i)$ to $ranked\_clients$
6: end for
7: sort $ranked\_clients$ by $\varphi_i$ in descending order
8: for all $(\varphi_i, i) \in ranked\_clients$ do
9:     if $\forall\, d \in \mathcal{D}:\ current\_qty[d] \geq r_d$ then
10:        break
11:    end if
12:    $new\_classes \leftarrow$ Classes($i$) $\setminus$ $covered\_classes$
13:    if $new\_classes = \emptyset\ \wedge\ \forall\, d \in$ Classes($i$) : $current\_qty[d] \geq r_d$ then
14:        continue
15:    end if
16:    if $total\_bid + \hat{P}_i > B_{NO}^{\mathrm{sel}}$ then
17:        continue
18:    end if
19:    append $i$ to $\mathcal{I}^\star$
20:    $total\_bid \leftarrow total\_bid + \hat{P}_i$
21:    $covered\_classes \leftarrow covered\_classes \cup$ Classes($i$)
22:    for all $d \in$ Classes($i$) do
23:        $current\_qty[d] \leftarrow current\_qty[d] + q_{i,d}$
24:    end for
25: end for
26: Return $\mathcal{I}^\star$
```

5.4.8 Computational Complexity

The client selection mechanism in Algorithm 9 is designed for practical deployment in FL environments, where the number of potential participants can scale to tens of thousands. Its design balances selection optimality with computational efficiency, ensuring that the process remains tractable even under stringent data coverage and quantity constraints.

The algorithm operates in three main phases: (1) computing each client's contribution score $\varphi_i = V_i(D_i, Q) + \alpha R_i - \beta \hat{P}_i$, which involves evaluating class coverage $C_i(D_i)$ and quantity sufficiency $S_i(Q)$ in $\mathcal{O}(n\,k)$; (2) ranking clients by their scores in $\mathcal{O}(n \log n)$; and (3) greedily scanning the sorted list to update cumulative coverage and quantities, which in the worst case is $\mathcal{O}(n\,k)$ but typically terminates far earlier once the per-class thresholds are satisfied. In realistic deployments, the number of required data classes k is much smaller than the number of clients n, and client data profiles are often sparse, making the ranking phase the dominant cost.

This yields an effective complexity that scales quasi-linearly with the number of clients, a substantial improvement over exhaustive search $\mathcal{O}(2^n)$, which is infeasible for moderate or large n. The selection phase is executed once before training begins, so its cost is amortized over all FL rounds without introducing per-round latency. This combination of algorithmic efficiency and engineering optimizations ensures the mechanism is theoretically scalable, as well as practically deployable in diverse FL scenarios.

5.4.9 Resilience to Attacks

In this section, we evaluate the resilience of our approach against common adversarial strategies such as free-riding and Sybil-based collusion.

1. **Free-Riders:** Clients attempting to contribute intermittently face an unfavorable balance: one dishonest round cancels several honest ones due to the punitive measure of PIPP. As a result, sporadic contributors can possibly fall below the selection criteria and are excluded, specially in tasks where reputation is important to the task owner and has a high reputation parameter (α).

2. **Sybil Attacks:** New identities initialize with $R = 0$ and require at least several consecutive honest rounds to gather a good selection reputation. The system enforces identity binding at the certificate or device level, preventing attackers from resetting reputations by simply rejoining. Detected misbehavior on any Sybil node triggers slashing across the identity cluster.

3. **Collusion:** Coordinated clients attempting to rotate updates or submit redundant gradients are detected through similarity metrics or shady participation patterns and penalized accordingly. As penalties are steeper than rewards, collusion does not result in sustainable selection.

Furthermore, a distinct potential threat exists at the auction stage, where clients may coordinate bids or reported data valuations to manipulate selection and reward outcomes. In our marketplace, this would require a coalition to align their submitted prices P_i and/or claimed data value $V_i(\mathcal{D}_i, Q)$ in order to collectively increase their ranking under the contribution score.

$$\varphi_i = V_i(D_i, Q) + \alpha R_i - \beta \hat{P}_i,$$

Collusion at this stage can manifest in several forms, such as bid shading, where coalition members submit artificially low prices to dominate selection; price inflation, where bids are elevated in expectation of making finding good clients harder; or valuation manipulation, where clients exaggerate their coverage of underrepresented data classes to boost V_i.

The mechanism design places several constraints that inherently limit the profitability of such strategies:

1. **Score–Price Coupling:** In our scoring rule, higher bids strictly reduce the client's score through the term $-\beta \hat{P}_i$, which decreases their likelihood of being selected. Under the critical-value payment rule, increasing one's bid does not increase the payment, winners always receive their critical bid, which is independent of their reported price as long as they remain below the threshold. Therefore, neither an individual client nor a coalition benefits from inflating bids: doing so only reduces their chances of selection, while providing no monetary advantage. Collusion based on price inflation becomes self-defeating unless the coalition controls irreplaceable data (e.g., rare classes or scarce quantities), in which case their market power comes from data scarcity, not strategic price manipulation.

2. **Coverage-Constrained Selection:** The greedy selection process enforces class coverage and quantity sufficiency constraints. Coalitions that do not jointly satisfy all uncovered data requirements cannot monopolize the auction outcome, even if their bids are strategically aligned.

3. **Persistent Reputation Effects:** Overstated utility claims that lead to low-quality updates are penalized via the PIPP mechanism in future tasks, making short-term auction manipulation unattractive for clients aiming to maintain long-term selection viability.

Overall, the interplay of price–score coupling, strict coverage constraints, and persistent reputational incentives makes sustained, profitable auction-stage collusion difficult to achieve in the proposed marketplace, while preserving the competitive dynamics necessary for high-quality model training.

5.5 Experimental Results

In this section, we evaluate the practical and statistical merits of our proposed FL marketplace through a comprehensive suite of simulations. The evaluation is conducted with a client pool of 500 devices. Unless otherwise stated, for empirical validation we rely on benchmark datasets; FashionMNIST and K-MNIST; partitioned in a non-IID manner using Dirichlet sampling. Our simulations are conducted against multiple baselines and state-of-the-art mechanisms, including the reverse-auction-based RRAFL [111] and the fairness-driven FAIR [77]. RRAFL operates through a reverse auction where clients are prioritized by their cost-to-reputation ratio, while FAIR normalizes data quality relative to cost to promote equitable allocation. Additionally, we benchmark the Naïve Greedy approach, which admits clients without regard for coverage, serving as a useful lower bound, as well as the baseline without incentives. For reputation, we employ a hybrid strategy for its assignment to the clients, and combine different reputations signals: some are parameterized by data diversity and quantity to provide reproducibility and transparent evaluation, while others are based on factors such as historical participation quality, task success rates, and client reliability. By allowing both data-driven and behavior-driven reputations to coexist within the same simulated setup, our framework captures a broader spectrum of reputation formation, reflecting how real federated marketplaces would integrate different dimensions when establishing trust. Furthermore, we distinguish between a planning budget used during client selection and the settlement payments induced by the critical-value payment rule. The planning budget is enforced by the heuristic selection algorithm using reported bids, while final settlement payments are computed ex post using critical values to ensure incentive compatibility. As a result, settlement expenditure may exceed the planning budget; this difference reflects the cost of enforcing truthful bidding.

5.5.1 Computational Feasibility

Before analyzing learning quality, we must certify that our approach can operate at the scale demanded by modern FL deployments. We therefore benchmark our greedy selector against the exhaustive search that enumerates every admissible client subset, treating the latter as a tight; yet impractical; upper bound on solution quality. In both cases, the underlying task is to assemble a feasible cohort of clients that collectively satisfy the task owner's per-class data quotas.

Let n denote the size of the candidate pool. The exhaustive strategy incurs combinatorial cost

$$T_{\text{exh}}(n) = \sum_{k=1}^{n} \binom{n}{k} = 2^n - 1 = \mathcal{O}\big(2^n\big),$$

whereas our selector leverages a priority queue to obtain

$$T_{\text{greedy}}(n) = \mathcal{O}\big(n \log n\big),$$

yielding an exponential–versus–quasi-linear gap.

We synthesize five progressively larger environments (Table 5.1) by varying the number of devices, the cardinality of the label space, and the per-class sample quota. Each device advertises a random multi-set of classes, where the number of samples it holds for each class is drawn from a Poisson($\lambda = 20$) distribution. In this scalability study, the budget constraint is intentionally non-restrictive, as our objective is to isolate and measure the runtime required to satisfy all class quotas. For instance, in Env-1 the task requires collecting 1,000 samples for each of 100 classes from a pool of 10,000 clients.

Table 5.1 Synthetic environments used for the scalability study.

Environment	**Devices**	**Classes**	**Min. Samples / Class**
Env-1	10 000	100	1 000
Env-2	20 000	200	2 000
Env-3	30 000	300	3 000
Env-4	50 000	400	4 000
Env-5	100 000	500	5 000

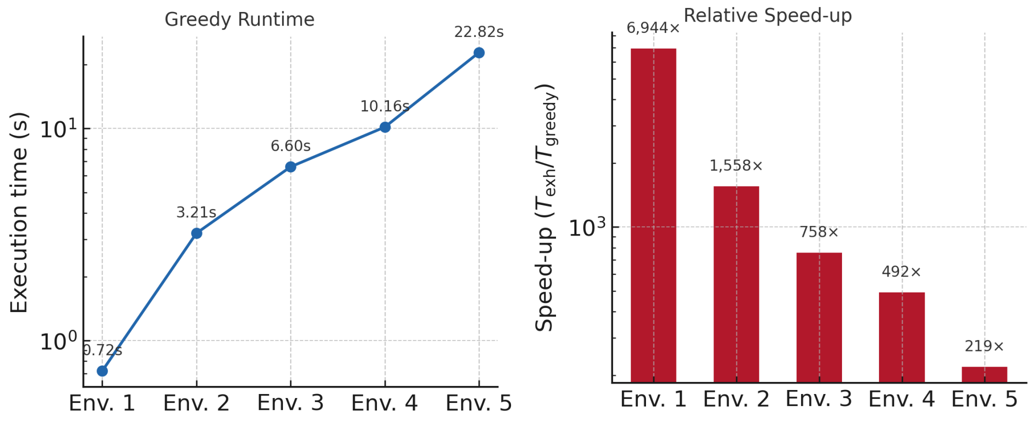


Figure 5.1 Wall-clock execution time and relative speed-up: greedy vs. exhaustive search.

Figure 5.1 highlights wall-clock execution time. The greedy selector exhibits near-linear growth, completing the 100 k-device scenario in just 22.8 s, whereas the exhaustive approach breaches 5 000 s even in the smallest setting. In terms of efficiency, our method achieves around 7×10^3 speed-up in Env-1 and maintains a $> 2 \times 10^2$ advantage at 100 k devices. Memory consumption also remains practical: peak RAM usage stays below 1 GB (0.9 GB at 100 k devices) due to streaming evaluation, while exhaustive enumeration exceeds 16 GB once $n > 20\,000$. The best-fit slope of execution time corresponds to only 0.23 ms per additional candidate, suggesting that the approach can comfortably accommodate tens of thousands of clients within a typical FL task. Collectively, these findings confirm that the proposed method operates within realistic latency and memory constraints without requiring specialized hardware, thereby eliminating the exponential bottleneck of exhaustive search and enabling subsequent accuracy-focused experiments to be conducted at realistic scale.

5.5.2 Evaluation Across Different Market Conditions

To systematically assess the effectiveness of our approach, we evaluate how it performs in terms of cohort selection and global model accuracy under a set of controlled market scenarios against different baselines. These scenarios are designed to reflect diverse conditions in federated marketplaces by varying key dimensions simultaneously, such as the budget, the degree of data availability, and the distribution of client reputations. We structure the evaluation around two representative settings: (i) scarce planning budget with reduced reliability, where the budget is capped at 1200 AU, resources are limited, data is highly skewed, and low-reputation participants dominate; and (ii) moderate planning budget with a balanced market, where the budget is capped at 1600 AU, resources are sufficient, heterogeneity is moderate, and participant quality is more evenly distributed. In both cases, the task owner requires five specific classes with the following quotas and weights: Class 0 with 900 samples (weight 1.0), Class 1 with 900 samples (weight 1.0), Class 4 with 750 samples (weight 1.5), Class 8 with 150 samples (weight 3.0), and Class 9 with 600 samples (weight 1.5).

On the FashionMNIST dataset (Figure 5.2), our approach demonstrates clear superiority under both scarcity and balance. In the scarce budget with reduced reliability scenario, it achieves 71.4% accuracy with only 38 clients, while FAIR follows with 67.5% using 44 clients, RRAFL drops to 64.5% with 66 clients, and Naïve Greedy lags behind at 52.1% despite including 103 clients. This illustrates how inclusion of many low-reputation participants severely undermines aggregation if the quality and the relevance of the owned data is neglected, while our selective mechanism maximizes contribution per client. Under the moderate budget with a balanced market, accuracy across methods converges, with our approach leading at 83.0% using 53 clients, FAIR close at 80.0% with 55 clients, RRAFL at 79.1%

with 86 clients, and Naïve Greedy at 77.3% with 114 clients. This convergence shows that fairness- and reliability-oriented schemes lose their disadvantages when resources are sufficient and the market is balanced, yet our approach continues to achieve the best accuracy with fewer clients, underscoring its efficiency.

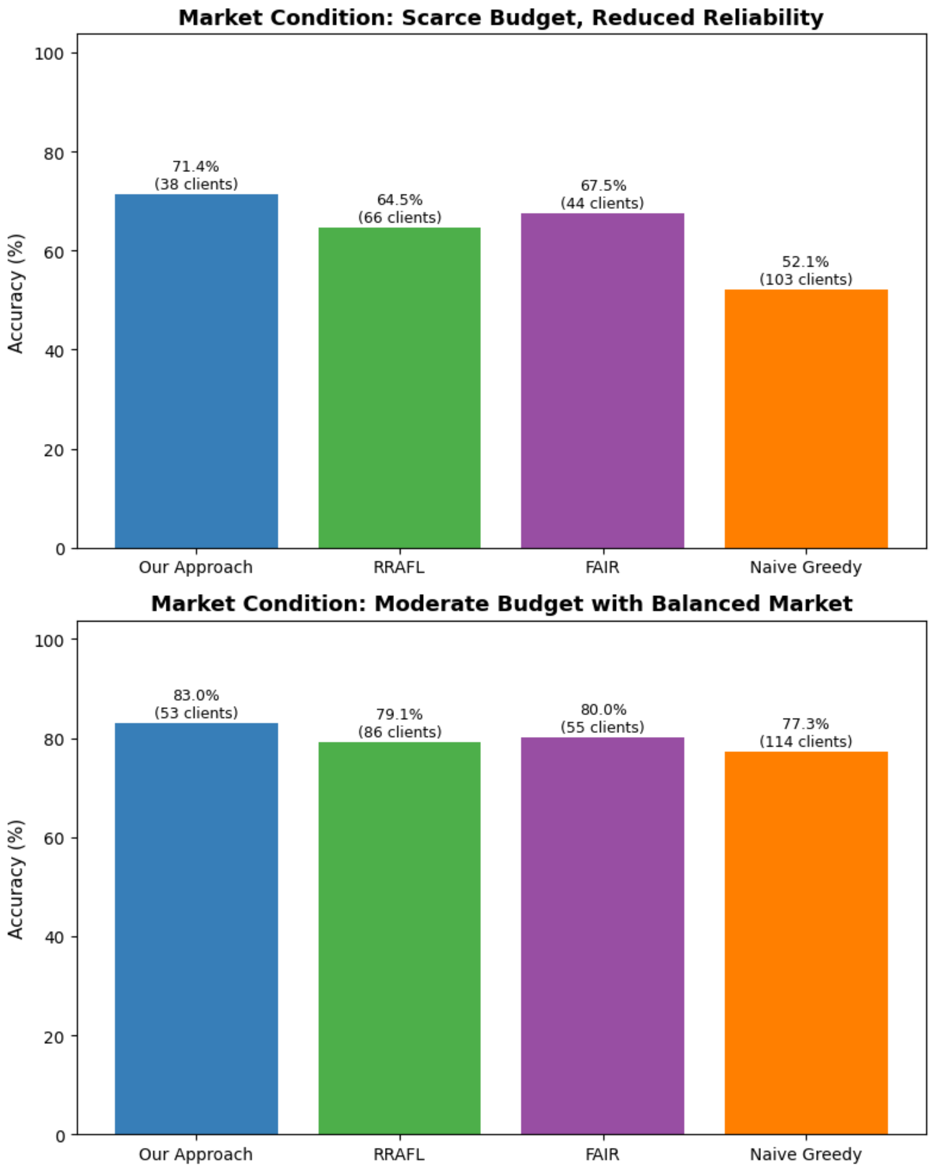


Figure 5.2 Accuracy and number of selected clients across different marketplace conditions under FashionMNIST.

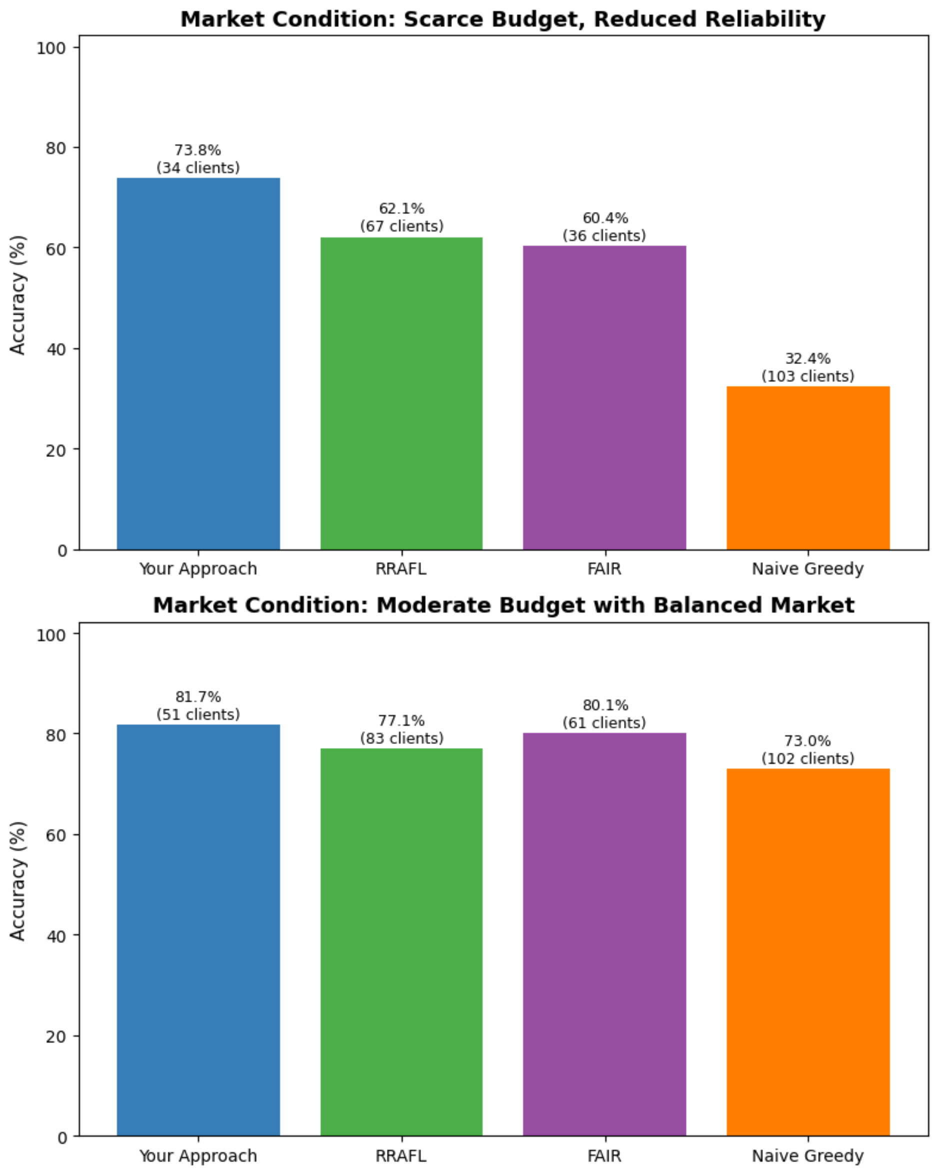


Figure 5.3 Accuracy and number of selected clients across different marketplace conditions under K-MNIST.

The results on the K-MNIST dataset (Figure 5.3) reinforce the strength of our approach, particularly under harsher conditions. In the scarce budget with reduced reliability setting, our mechanism secures 73.8% accuracy with only 34 clients, outperforming RRAFL (62.1% with 67 clients), FAIR (60.4% with 36 clients), and Naïve Greedy (32.4% with 103 clients). Here the weaknesses of alternative methods are more pronounced: Naïve Greedy suffers most from indiscriminate selection, while RRAFL and FAIR dilute effectiveness through excessive reliance on reliability weighting and fairness constraints. By selectively prioritizing reliable and data-rich contributors, our method preserves efficiency and sustains strong accuracy under scarcity. With a moderate budget and balanced market, performance levels rise and differences narrow, but our approach maintains the lead at 81.7% with 51 clients, compared to FAIR at 80.1% with 61 clients, RRAFL at 77.1% with 83 clients, and Naïve Greedy at 73.0% with 102 clients. These results confirm that while competing approaches improve when resources are sufficient and heterogeneity is reduced, our mechanism consistently secures higher accuracy with fewer participants, maintaining a clear efficiency advantage across market conditions.

5.5.3 Privacy Budgets via Differential Privacy (DP)

Formal privacy guarantees can be further enhanced through the integration of techniques like Differential Privacy (DP). DP offers a rigorous mathematical framework for quantifying and limiting the disclosure of individual client information from the aggregated model. This section analyzes how our marketplace framework can be extended to incorporate client-side DP and evaluates the resulting trade-off between the strength of the privacy guarantee and the utility of the final model, measured by its accuracy. The amount of noise, is determined by the desired privacy budget, denoted by epsilon (ϵ). Larger ϵ corresponds to weaker guarantees (less noise).

Table 5.2 Impact of Differential Privacy on Model Accuracy

Metric	$\varepsilon = 20$	$\varepsilon = 25$
Accuracy without DP (%)	85.27	85.27
Accuracy with DP (%)	81.67	82.06
Accuracy Loss (%)	3.60	3.09

The results in Table 5.2 illustrate the inherent trade-off between privacy and model utility in our framework. We conduct the simulation in a controlled environment with 500 candidate clients under a planning budget of 1800 AU, comparing performance without privacy guarantees against two differential privacy settings with distinct privacy budgets, relying on the same requirements of the experiment in section 5.5.2. The results showcase that, introducing DP reduces the final model accuracy due to the injected noise, with a stronger privacy guarantee (smaller ϵ) incurring a larger performance drop. For $\epsilon = 20$, accuracy decreased by 3.60%, while for $\epsilon = 25$, the drop was reduced to 3.09%. This highlights that higher ϵ values (weaker privacy guarantees) preserve more model utility, whereas lower ϵ values (stronger guarantees) offer better protection at the cost of accuracy. In practice, selecting an appropriate ϵ depends on the application's tolerance for performance loss relative to the required privacy level.

5.5.4 Impact of the PIPP Reputation Mechanism

To isolate the contribution of the PIPP reputation protocol, we perform a pre-training phase in which our marketplace executes five independent tasks in a synthetic environment. This pre-training phase is used solely to let clients accumulate reputation scores. During those runs the greedy selector records individual reliability behavior, so that, by the sixth task, every client possesses a non-zero reputation vector $R_i \in [0, 1]$. We then run another task under two mutually exclusive settings: (1) No-PIPP, and (2) PIPP-Enabled. All other hyper-parameters are held constant, under a planning budget of 1200 AU.

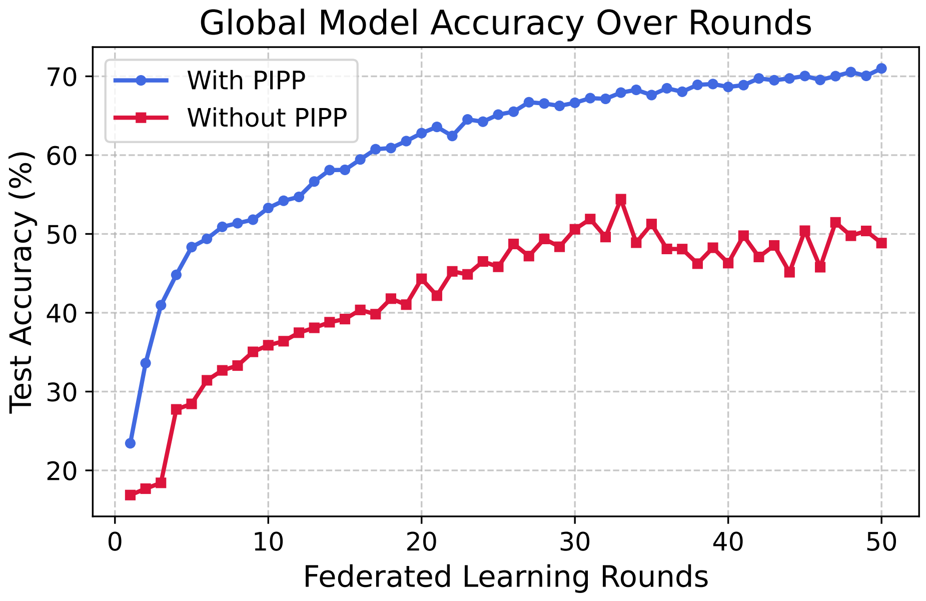


Figure 5.4 Impact on Model Accuracy in Scenarios Utilizing and Not Utilizing PIPP

Figure 5.4 reveals a statistically stable advantage for the reputation-aware approach relying on PIPP. The PIPP enabled task surpasses the 60 % accuracy threshold by round 20, whereas the No-PIPP task does not reach 50 % until round 30. After 50 communication rounds the accuracy is 71.00% with PIPP versus 48.84% without; a difference of 22.2%.

These outcomes substantiate the intuition that trust-weighted bids steer the marketplace towards clients who are simultaneously data-valuable and behaviorally honest, and reliable, thereby accelerating convergence and lifting the ceiling on attainable performance. In practical deployments, this translates into fewer rounds to reach a target accuracy and a more resilient global model in the presence of heterogeneous, potentially adversarial participants.

5.5.5 Analysis of Initial Reputation Priors and Marketplace Fairness

The initialization of newcomer reputation, $R_i^{(0)}$, is a critical design parameter in federated marketplaces, shaping the balance between resilience against Sybil attacks and fairness for legitimate entrants. Our framework defaults to a conservative zero-start reputation $R_i^{(0)} = 0$, which maximizes security by requiring all identities to gradually accumulate credibility through demonstrated contributions. To study the broader systemic effects of different initialization choices, we simulated 400 established clients and 100 newcomers under four priors, $R_i^{(0)} \in \{0.0, 0.3, 0.5, 1.0\}$, with system performance tracked across 10 consecutive tasks under environment conditions similar to section 5.5.2's scarce budget with reduced reliability scenario using the K-MNIST dataset. The results are shown in Table 5.3.

Table 5.3 Impact of Initial Reputation Priors on Newcomer Participation and Model Accuracy

Positive Welcome Prior (R)	**Final Model Accuracy (%)**
0.0	73.14
0.3	73.75
0.5	77.85
1.0	79.93

The outcomes reveal a clear upward trend in model accuracy as the initialization prior increases. Accuracy improves from 73.14% at $R = 0.0$ to 79.93% at $R = 1.0$, with intermediate priors yielding incremental gains. These improvements reflect an indirect mechanism: higher priors shift client rankings and displace weaker incumbents, thereby intensifying competition and producing cohorts of greater overall quality.

Taken together, the findings highlight two complementary insights. First, a strict zero-start policy provides the strongest protection against Sybil attacks but may result in more conservative overall performance. Second, positive initialization priors, when carefully calibrated, can act as systemic levers, indirectly enhancing model quality without undermining security guarantees. This suggests that reputation initialization should be treated not merely as a security safeguard but as a tunable design choice with measurable impact on utility. As a promising direction for future work, hybrid or adaptive strategies, for example, context-aware priors that vary with client features or task requirements, could jointly achieve resilience against adversarial newcomers while also unlocking efficiency gains in large-scale federated marketplaces.

5.5.6 Optimality-Gap Study (Greedy vs. Exhaustive Selection)

To quantify how closely the proposed greedy selector approaches the true optimum, we conducted a small-scale experiment with 20 clients competing for slots under a 70 AU budget cap. Each client owned a non-IID partition of a synthetic dataset with 6,000 samples, 20 features, and 3 classes, incurred a private cost $P_i \in [10, 40]$ AU, was assigned a reputation $R_i \sim \mathcal{U}(0.20, 1.00)$, and contributed data value V_i via our DCVF metric. A valid subset required satisfying both the budget constraint and a minimum of 300 samples per class.

We compare our greedy strategy against the exhaustive method that enumerates all feasible client combinations and chooses the utility-maximizing set. Because exhaustive enumeration scales as $\mathcal{O}(2^n)$, this experiment is restricted to a small client pool and serves as a tight benchmark for validating the practical quality of the heuristic.

Table 5.4 reports the global test accuracy and total utility for each approach.

Table 5.4 Optimality-gap experiment results

Approach	**Global Model Accuracy**
Greedy (heuristic)	91.00%
Optimal (exhaustive)	**91.25%**

The results highlight that the exhaustive method yields only a 0.25 percentage point accuracy improvement despite requiring significant computational overhead. These results indicate that our greedy strategy achieves over 99% of the optimal utility while remaining effective, this highlights the practical optimality of our selection mechanism.

5.6 Conclusion

In this chapter, we discussed a novel three-layer FL marketplace architecture that explicitly distinguishes between task owners, network owners, and clients. Our framework enables clients to autonomously bid for participation and earn economic incentives using a reverse auction mechanism combined with a proprietary, internal, perpetual, and punitive (PIPP) reputation system. We demonstrated through extensive experiments that our method achieves better accuracy with fewer participating clients compared to the state-of-the-art, while maintaining computational efficiency.

CHAPTER 6 ARTICLE 4: A TWO-LEVEL DIRICHLET FRAMEWORK FOR HETEROGENEOUS FEDERATED NETWORK

Accepted for publication in IEEE Transactions on Network Science and Engineering on August 5, 2025. Authors: Sarhad Arisdakessian, Osama Wehbi, Omar Abdul Wahab; Azzam Mourad, Hadi Otrok, Mohsen Guizani.

Building on the foundations laid in the preceding chapters, where we established the motivation for a federated learning marketplace and analyzed its key challenges, we now turn to one of the most fundamental obstacles to effective FL in practice: the prevalence of heterogeneous and non-IID data across clients and realistic methods to generate non-IID distributions for simulation and experiments. This chapter addresses this challenge by situating our proposed solution within the broader landscape of FL research. Interest in FL has surged in recent years as evidenced by a significant rise in the number of relevant publications [112], as well as broader adoption across sectors such as healthcare [113, 114, 115, 116, 117], Industrial IoT (IIoT) [118, 119, 120], and finance [121, 122, 123]. Many of this works tackle and try to alleviate the negative effects of non-IID data in real-world FL environments [124, 125, 126, 127, 128, 129, 130, 131, 132]. At its core, FL depends on each client's resources, particularly their locally available data, to power distributed training. In practical scenarios, however, data quantity and quality vary greatly among clients, resulting in non-Identically Independently Distributed (non-IID) data [111, 133]. This data heterogeneity [1] may stem from various factors, such as users generating data under different contexts, devices, or demographic groups. Common manifestations include differences in class label presence (label skew), variations in feature characteristics for the same class (feature skew), and unequal dataset sizes across clients (quantity skew). These types of non-IIDness introduce significant challenges and slows model convergence [23, 134, 135], motivating the need for realistic partitioning techniques. A formal breakdown of these heterogeneity types is provided in Section 6.2.

Compounding this issue, researchers often lack access to real-world deployments and data, thus resorting to simulations. To emulate real-world conditions and rigorously evaluate FL methods under non-IID settings, researchers frequently employ well-known centralized datasets (e.g., MNIST with its variants, CIFAR, Shakespeare, Adult Income, etc.) and partition them in ways that reflect diverse client distributions. A common partitioning approach involves applying the Dirichlet distribution [136] to manipulate label proportions. While this method improves upon simple IID sampling by modeling label imbalance, it fails to capture

[1] Hereafter, we use the terms "non-IID" and "data heterogeneity" interchangeably

deeper heterogeneity, such as variations in handwriting styles within the same digit class or different accents for the same language label.

In this work, we propose a feature-aware distribution technique that both builds on and extends the traditional Dirichlet-based partitioning commonly used in FL. By allowing subpopulations with shared semantic or stylistic characteristics to be skewed in distinct ways across client devices, our approach more accurately captures the rich, multi-dimensional data diversity often found in real-world data distributions. This added nuance surpasses the limitations of label-only Dirichlet methods and provides service managers and system architects with a modular and parameterized framework for simulating non-IID distributions. As a result, the proposed technique more reliably stress-tests FL deployments, ensuring that performance and reliability objectives can be upheld in heterogeneous and large-scale, decentralized, and distributed settings.

In real federated settings, different clients naturally collect data with varying feature, label distributions, or both. For instance, phone cameras may capture images in different lighting conditions or user contexts; hospitals might serve diverse patient populations with different disease prevalence. While publicly available datasets with client-level heterogeneity exist such as [137], studies still rely on synthetic data partitioning to systematically evaluate FL methods under non-IID conditions. One widely adopted technique involves Dirichlet(α) sampling over class labels, where a smaller (α) value yields more skewed label distributions. Such single-level Dirichlet splits, while straightforward and flexible, capture only label imbalance, failing to account for the feature-level variations within a class. In many real-world scenarios, feature-space differences, such as demographic or device-based discrepancies, can cause subpopulations to differ significantly within the same label category (e.g., different handwriting styles for the digit 8, or domain shifts in text corpora). This mismatch between synthetic label skew and genuine multi-dimensional non-IIDness can lead to overly optimistic or misleading results in FL based experiments.

For these reasons, label-only Dirichlet partitioning has drawn criticisms for its oversimplified view of heterogeneity. Furthermore, the diversity of real-world non-IID data is often underrepresented. In particular, traditional Dirichlet distribution falls short in:

- **Feature-Level Complexity**: Real data heterogeneity isn't only about having unbalanced class proportions. Two clients might both have the same set of classes but still differ significantly in features (e.g., handwriting styles, camera sensor differences).

- **Within-Class Variability**: Even for a single class (e.g., 'cat' images), real data might span subpopulations (e.g., different breeds, angles, lighting) that get distributed differently across devices. A single-level Dirichlet partition on class labels can't capture such subtleties.

These limitations motivate the need for a more expressive partitioning framework that captures both feature- and label-level heterogeneity to better reflect the complexity of real-world federated data distributions.

6.1 Traditional Dirichlet Distribution in FL

In this section, we focus on the use of Dirichlet partitioning in the context of FL where it is often used to create 'non-IID' partitions of labeled datasets and simulating heterogeneous data across clients. Specifically, in such environments, each client is assigned a random proportion of each class, and these proportions are drawn from a Dirichlet distribution.

Mathematically, the Dirichlet distribution is a probability distribution over a $(K-1)$-dimensional simplex. In particular, it is a distribution over k-component probability vectors $\theta = (\theta_1, \ldots, \theta_k)$ where:

$$\theta_i \geq 0, \sum_{i=1}^{k} \theta_i = 1 \tag{6.1}$$

The Dirichlet distribution with parameter vector $\alpha = (\alpha_1, \ldots, \alpha_k)$, where $\alpha_i > 0$ for all i, has a Probability Density Function (PDF) represented by:

$$Dir(\theta|\alpha) = \frac{1}{B(\alpha)} \prod_{i=1}^{k} \theta_i^{\alpha_i - 1} \tag{6.2}$$

Where:

$$B(\alpha) = \frac{\prod_{i=1}^{k} \Gamma(\alpha_i)}{\Gamma\left(\sum_{i=1}^{k} \alpha_i\right)} \tag{6.3}$$

Such that:

- $\Gamma(.)$ is the Gamma function (generalization of factorial)
- B(α) is the multivariate Beta function (normalizing constant)
- $\theta_i \geq 0$ and $\sum_{i=1}^{k} \theta_i = 1$ are the support

Based on the above, the value of α_i plays a very important role in the distribution of data, as it serves as a concentration parameter, such that:

1. $\alpha_i < 1$: Tends to produce more extreme draws near the corners of the simplex where one dimension dominates.
2. $\alpha_i = 1$ for all i: This is the uniform distribution over the simplex.
3. $\alpha_i > 1$: Tends to concentrate draws around the center of the simplex and provides more even distributions among the categories.
4. $\alpha = (\alpha_1, \ldots, \alpha_k)$ has all the components the same ($\alpha_1 = \cdots = \alpha_k = \alpha_0$), which is a special case known as the 'Symmetric Dirichlet'. This formulation is often used for simplicity in probability modeling, as it ensures that all categories are assigned equal prior concentration, leading to more balanced but still flexible distributions.

It is important to note here that having α=1 does not mean a distribution that is close to IID; rather this makes the Dirichlet distribution uniform over all possible label proportion vectors, meaning each configuration is equally likely. The distribution still randomly allocates the class samples to clients. Some clients may still receive more or fewer samples of certain classes. Hence, even though α=1 tilts toward a 'balanced' expected distribution, the actual draw can produce moderate label imbalances, just less extreme than from α <1.

6.2 Two-Level hierarchical Dirichlet Distribution

The heterogeneity in FL can manifest in various forms, which impact model training and performance [7]. Formally, for clients $i \neq j$, heterogeneity exists if:

$$\mathbb{P}(x, y \mid D_i) \neq \mathbb{P}(x, y \mid D_j)$$

1. **Label Skew**: Certain clients may predominantly contain samples from specific classes. For example, one client might mostly have images of cats, while another has mostly images of dogs in an image classification task.

$$\mathbb{P}(y \mid D_i) \neq \mathbb{P}(y \mid D_j)$$

2. **Feature Skew**: Even when clients share similar distributions of class labels, the underlying features of the data may differ significantly. This can occur due to variations

in styles, environments, or sensor modalities; for instance, different handwriting styles in a digit recognition task.

$$\mathbb{P}(x \mid y, D_i) \neq \mathbb{P}(x \mid y, D_j)$$

3. **Quantity Skew**: The amount of data available per client may vary widely, with some clients contributing large datasets while others possess only a small number of samples.

$$|D_i| \neq |D_j|$$

Hereafter, we discuss in detail our proposed framework that enhances the traditional Dirichlet-based partitioning method to generate a more realistic, parameterized, and tuneable non-IID data distribution for FL scenarios. Unlike conventional techniques that primarily focus on label-based partitioning, our method integrates feature-space clustering, and a hierarchical sampling mechanism to provide a more nuanced and representative data distribution. By leveraging this unified framework, we systematically address all three major forms of data heterogeneity: (1) label skew, (2) feature skew, and (3) quantity skew, using a single technique. This approach ensures that researchers and the broader scientific community can simulate real-world FL conditions more effectively, enabling more robust evaluations of FL algorithms.

6.2.1 Proposed Methodology

In this section, we discuss the different steps that constitute our proposed framework.

Step 1: Feature Extraction

The success of the clustering step in our partitioning framework relies on the availability of semantically meaningful embeddings that reflect the intrinsic structure of the dataset. To ensure this, we employ feature extractors that are pretrained on large-scale, general-purpose datasets, and are used in a frozen form; i.e., without additional fine-tuning or supervision on the federated task. This allows the feature extraction process to remain completely decoupled from the downstream FL objective. The choice of feature extractor is modular and depends on the modality of the input data. For visual data, typically convolutional neural networks such as ResNet-18 or VGG-16 can be used, whose hierarchical structure captures both low-level and high-level visual patterns. For textual data, transformer-based language models like BERT or RoBERTa can be utilized. These pretrained models are widely available and can be applied directly to input data using standard inference pipelines without requiring task-specific supervision.

This design offers two key advantages: (1) it enables the clustering to occur in a high-dimensional semantic space where subpopulation structure can be more effectively captured, and (2) it ensures that the partitioning process can generalize across domains without access to labels or task-specific feedback.

Mathematically, the feature extraction process can be expressed as follows:

$$z_i = f_\theta(x_i) \tag{6.4}$$

where:

- x_i represents the raw input data instance,
- $f_\theta(\cdot)$ denotes the feature extraction function parameterized by θ, which corresponds to the pre-trained or task-specific model,
- z_i is the extracted feature vector representing x_i in a d-dimensional latent space:

$$z_i \in \mathbb{R}^d \tag{6.5}$$

where d is the dimensionality of the feature space.

The extracted features serve as the foundation for the subsequent clustering step, allowing us to structure the data in a way that reflects deeper similarities beyond class labels. This step is critical in enabling an advanced non-IID partitioning strategy, as it ensures that data points are grouped based on their characteristics, leading to a more meaningful and challenging FL data distribution.

Step 2: Clustering of the Feature Space

After obtaining high-dimensional feature embeddings from the dataset, the next step involves performing unsupervised clustering to partition the extracted feature space into meaningful sub-populations. The goal of this step is to capture intrinsic structures within the data beyond class labels, ensuring that data instances with similar underlying characteristics are grouped together. These clusters serve as an intermediate representation that facilitates the generation of a more realistic and hierarchical non-IID data distribution. The selection of an appropriate clustering method depends on the nature of the dataset and the complexity of its feature distributions, but possible options include K-Means, and DBScan.

Formally, given a dataset of N feature-extracted instances $\{z_i\}_{i=1}^{N}$, where:

$$z_i = f_\theta(x_i), \quad z_i \in \mathbb{R}^d \tag{6.6}$$

we apply a clustering algorithm $\mathcal{C}$ to assign each instance to one of K clusters:

$$k_i = \mathcal{C}(z_i), \quad k_i \in \{1, \ldots, K\} \tag{6.7}$$

where k_i represents the cluster index assigned to z_i.

The clustering step partitions the feature space into K clusters, with each cluster representing a coherent sub-population of data points. The output is a cluster label:

$$k_i \in \{1, \ldots, K\}, \quad \forall x_i \in \mathcal{D} \tag{6.8}$$

Where x_i is a data sample that belongs to the dataset $\mathcal{D}$. These clusters serve as the foundation for the sampling step which follows, where the distribution of clusters across clients is controlled to generate stronger non-IID characteristics.

We note that in our implementation, we use K-means clustering due to its simplicity and, importantly, its ability to explicitly control the number of clusters (K). This tunability allows us to directly modulate the granularity of simulated subpopulations, providing a structured way to influence the feature-level heterogeneity. However, our framework is agnostic to the clustering algorithm used and can be extended to methods such as DBScan, depending on the geometry or density of the feature space. In practice, for algorithms like DBScan, satisfactory clustering results often require careful parameter selection or pre-processing steps, such as performing a parameter sweep to identify suitable values for the neighborhood radius (epsilon) and minimum samples. The optimal parameter choices can vary across datasets, and additional attention is often necessary to ensure that the resulting clusters achieve the intended balance of heterogeneity. We leave a deeper comparative study of clustering choices, as well as the systematic exploration of clustering parameters, as an important direction for future work.

Step 3: Two-Level Hierarchical Non-IID Distribution

This step constitutes the core innovation of our approach, extending beyond the conventional Dirichlet-based label partitioning to generate a more realistic non-IID data distribution. Unlike traditional data partitioning methods that apply a Dirichlet distribution directly to class labels, our approach introduces a hierarchical sampling strategy that first distributes data at the cluster level, followed by a class-level refinement. This ensures that data heterogeneity is controlled at multiple levels, mimicking real-world scenarios where differences arise both in feature-space distribution and class distribution.

Step 3.1: Cluster-Level Dirichlet Sampling

The first part of the non-IID partitioning is performed at the cluster level. Given a set of feature-extracted data points $\{z_i\}_{i=1}^{N}$ that have been grouped into K clusters via clustering (6.2.1), we distribute these clusters across M clients. This is achieved by sampling a vector of cluster mixing proportions $\pi^{(k)} = (\pi_1^{(k)}, \ldots, \pi_M^{(k)})$ for each cluster k from a Dirichlet distribution:

$$\pi^{(k)} \sim Dir(\alpha^{(k)}) \tag{6.9}$$

where:

- $\alpha^{(k)}$ is a hyperparameter vector controlling the degree of non-IIDness in the distribution of cluster k across M clients.
- The dimensionality of $\pi^{(k)}$ is M, ensuring that each client receives a proportion of the data from cluster k.
- $\pi_j^{(k)}$ represents the proportion of cluster k's data assigned to client j.

Thus, the probability of a data point x_i belonging to cluster k being assigned to client j follows:

$$p(client = j \mid x_i \in k) = \pi_j^{(k)} \tag{6.10}$$

Step 3.2: Class-Level Refinement

Since each cluster k contains instances from multiple class labels, a second level of distribution is applied to refine the allocation of data points at the class level. This ensures that within each cluster, different classes are still heterogeneously distributed across clients.

To achieve this, we sample another Dirichlet distribution for each class c within a cluster k, yielding a sub-proportion vector:

$$\phi^{(k,c)} = (\phi_1^{(k,c)}, \dots, \phi_M^{(k,c)}) \sim Dir(\beta^{(k,c)}) \tag{6.11}$$

where:

- $\beta^{(k,c)}$ is the Dirichlet concentration parameter that dictates how class c is partitioned across clients within cluster k.
- $\phi_j^{(k,c)}$ represents the proportion of class c's instances from cluster k assigned to client j.
- Lower values of $\beta^{(k,c)}$ produce more imbalanced allocations, increasing the label skew among clients.

Thus, the probability of a data point x_i in cluster k with label c being assigned to client j follows:

$$p(client = j \mid x_i \in k, y_i = c) = \phi_j^{(k,c)} \tag{6.12}$$

Step 3.3: Final Data Assignment

To determine the final assignment probability for each data point x_i in cluster k with label c, we combine the probabilities obtained from both the cluster-level and class-level Dirichlet distributions:

$$p(client = j \mid x_i) = \pi_j^{(k)} \times \phi_j^{(k,c)} \tag{6.13}$$

A normalization step can be applied to ensure that:

$$\sum_{j=1}^{M} p(client = j \mid x_i) = 1 \tag{6.14}$$

Once the normalized probability vector p is obtained, the client assignment is sampled using a categorical distribution,

$$j \sim \mathrm{Cat}(p). \tag{6.15}$$

This ensures that each data point is assigned to a client proportionally to its cluster-level and class-level preferences, while maintaining a valid probability mass function required by the categorical sampler.

The proposed method ensures that data distributions in FL settings exhibit both feature-space heterogeneity (via clustering) and class-label heterogeneity (via hierarchical Dirichlet sampling). This methodology is significantly more flexible and realistic than traditional Dirichlet-based label partitioning, as it captures:

- **Feature Skew:** Clients receive data from distinct clusters, ensuring that even instances from the same class may exhibit different feature distributions.
- **Label Skew:** Within each cluster, different classes are allocated unevenly across clients, leading to an imbalanced distribution of labels.
- **Quantity Skew:** The hierarchical sampling allows control over how much data each client receives, leading to imbalanced dataset sizes per client.

6.2.2 Proposed Algorithm

In this section, we propose Algorithm 10, which consolidates the steps of our approach into a cohesive and comprehensive framework for generating non-IID data distributions in FL. By integrating feature extraction, clustering, and hierarchical Dirichlet sampling, the algorithm systematically partitions data across clients, ensuring a controlled and realistic degree of non-IIDness. This structured approach provides a clear and reproducible method for evaluating FL models under varying data heterogeneity conditions.

The algorithm's inputs are the dataset, number of clients, number of clusters, Dirichlet hyperparameters, as well as an optional minimum data point threshold τ for each client. This constraint is only applied if explicitly set by the user, and is optional. The outputs are the final partitions of the data. Step 1 introduces feature extraction, and lines $1-3$ form a loop extracting embeddings $\mathbf{z}_i$ for each sample x_i. In step 2 the clustering takes place, where lines $4-6$ cluster the embeddings, assign each sample to a cluster k_i, and record the index sets $\mathcal{C}_k$. Step 3.1 (cluster-level Dirichlet sampling), and lines $7-9$ loop over each cluster k to draw a Dirichlet vector $\boldsymbol{\pi}^{(k)}$. Then Step 3.2 introduces the class-level Dirichlet sampling within each cluster, with lines $10-15$ enumerating the classes in each cluster, sampling

Algorithm 10 Two-Level Dirichlet Partitioning for FL

```
Require: Labeled dataset $D = \{(x_i, y_i)\}_{i=1}^N$, number of clients $M$, number of clusters $K$,
    Dirichlet hyperparameters $\{\alpha^{(k)}\}_{k=1}^K$, $\{\beta^{(k,c)}\}$ for each cluster–class pair, minimum client
    size threshold $\tau$ (optional)
Ensure: Partition $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_M\}$, where $\mathcal{P}_j$ is set of data indices assigned to client $j$

    Step 1: Feature Extraction
 1: for $i \leftarrow 1$ $N$ do
 2:     $\mathbf{z}_i \leftarrow f_\theta(x_i)$
 3: end for

    Step 2: Clustering in Feature Space
 4: Cluster $\{\mathbf{z}_i\}_{i=1}^N$ into $K$ groups (e.g., K-Means)
 5: $k_i \leftarrow$ cluster assignment for sample $i$
 6: $\mathcal{C}_k \leftarrow \{\, i \mid k_i = k \,\}$ for $k = 1, \ldots, K$

    Step 3.1: Cluster-Level Dirichlet Sampling
 7: for $k \leftarrow 1$ $K$ do
 8:     $\boldsymbol{\pi}^{(k)} \sim \mathrm{Dir}\Big(\alpha^{(k)}\Big)$
 9: end for

    Step 3.2: Class-Level Dirichlet Sampling within Each Cluster
10: for $k \leftarrow 1$ $K$ do
11:     $C_k \leftarrow$ set of classes present in cluster $k$
12:     for all $c \in C_k$ do
13:         $\boldsymbol{\phi}^{(k,c)} \sim \mathrm{Dir}\Big(\beta^{(k,c)}\Big)$
14:     end for
15: end for

    Step 3.3: Data Assignment to Clients
16: for $i \leftarrow 1$ $N$ do
17:     $k \leftarrow k_i$;   $c \leftarrow y_i$
18:     $p_i(j) \leftarrow \pi_j^{(k)} \cdot \phi_j^{(k,c)} \quad \forall j \in \{1, \ldots, M\}$
19:     Normalize $p_i(\cdot)$ so that $\sum_{j=1}^M p_i(j) = 1$
20:     Sample $j_i \sim \mathrm{Cat}\Big(p_i(\cdot)\Big)$
21:     Assign sample $i$ to $\mathcal{P}_{j_i}$
22: end for

    (Optional) Step 3.4: Balancing Constraints
23: if $\exists j \,:\, |\mathcal{P}_j| < \tau$ then
24:     $U \leftarrow \{\, j \mid |\mathcal{P}_j| < \tau \,\}$
25:     $R \leftarrow \{\, k \mid |\mathcal{P}_k| \gg \tau \,\}$
26:     for all $j \in U$ do
27:         Transfer a small number of samples from clients in $R$ to $j$ until $|\mathcal{P}_j| \geq \tau$
28:     end for
29: end if

30: return $\mathcal{P}$
```

$\boldsymbol{\phi}^{(k,c)}$ for each class c, and finalizing the within-cluster distributions. Step 3.3 marks the data assignment process, and lines $16-22$ iterate over each sample to compute the product of cluster-level and class-level probabilities $(\pi_j^{(k_i)} \times \phi_j^{(k_i,y_i)})$, draw a client assignment, and place each sample i into $\mathcal{P}_{j_i}$. Step 3.4 introduces the optional step and lines $23-29$ check whether any client has fewer than τ samples; if so, a balancing procedure is triggered that transfers a small number of samples from clients with significantly larger partitions to meet the minimum threshold. Finally, line 30 returns the complete partition $\mathcal{P}$.

6.2.3 Computational Complexity

In this section we highlight the computational complexity of our approach and discuss its validity to use in real-world scenarios. In particular, our approach is divided into different phases, with each one adding a different computational layer to the partitioning process. Therefore, we analyze the computational complexity of each part and then provide the overall complexity of our approach and discuss its validity to be used by researchers.

Step 1: Feature Extraction

At this stage, each point of the N data points is transformed into a feature vector z_i. The cost of this is associated with the method or the model used for feature extracting. If a pre-trained neural model is used such as ResNet-18 or BERT for text, each forward pass has a complexity of around $O(D_{model})$. Passing it over N data points would yield $O(N \times D_{model})$. For text or tabular based data, if PCA is used, the cost is driven by the matrix factorization step, which depends on dimensionality F of any iterative solver, it would yield $O(N.F.min(N, F))$

Step 2: Clustering in Feature Space

Once each data point has a feature vector, they are clustered into K subpopulations. A common choice for clustering is K-means because of its parameterized ability to choose the number of output clusters which would give the user more control on the non-IIDness of the data distribution. Using K-means with I iterations, it costs about $O(I \times N \times K_{dim})$ where K_{dim} is the dimension of the feature vectors. For extremely larger datasets, mini-batch or approximate clustering can be used to reduce cost.

Step 3.1: Cluster Level Dirichlet

This steps samples once per cluster. $O(K)$ draws of a size-M Dirichlet vector. To generate each vector, it would yield a cost of $O(M)$.

Step 3.2: Within-Cluster Class-Level Dirichlet:

During this step, for each cluster k that contains classes $c1, c2, \ldots$, a vector is sampled per class with $O(M)$. Summing this accross all clusters, yields $O(K \times C_k) \times M$, where C_k is the number of classes in cluster k.

Step 3.3: Data Assignment

In this step, for each data point i, a final sample is computed from $\pi_j^{(k)} \times \phi_j^{(k,c)}$ with a complexity of $O(N)$.

Discussion:

Since our approach introduces additional steps, it naturally incurs some computational overhead. The most resource-intensive stages are Step 1 (Feature Extraction) and Step 2 (Clustering) due to the need for embedding generation and grouping data based on feature similarities. However, these computations remain feasible within a reasonable timeframe when working with moderate-sized, off-the-shelf datasets, which are commonly used in FL research. Importantly, since the data partitioning occurs before federated training begins, its complexity is significantly lower than that of the actual training process.

For datasets on the order of tens of thousands of samples, typical in standard image and text benchmarks, feature extraction remains manageable. In cases involving millions of samples, computational efficiency can be improved by using mini-batches or approximate clustering techniques like mini-batch K-means, reducing complexity to O(N × d). Additionally, embedding extraction can leverage batch processing and GPU acceleration, both of which are increasingly supported by modern deep learning frameworks, keeping runtime practical. Furthermore, partitioning is a one-time preprocessing step, meaning it does not contribute to the computational cost in subsequent FL training rounds. While our approach does introduce a higher initial computational cost compared to the traditional Dirichlet method, it remains viable for most FL research settings, particularly when the dataset size is reasonable. In return, it enables more realistic non-IID simulations, better reflecting real-world data heterogeneity, an advantage that outweighs the additional overhead.

6.3 Simulation and Experimental Results

In this section, we evaluate the versatility of our approach through simulations on diverse datasets and benchmarks. The goal is to highlight how our framework performs under varying scenarios and demonstrate its ability to emulate the complexities of real-world data distributions, offering a practical and consistent alternative to using actual private datasets. We refer to our approach across this section as ‘Hierarchical’.

Our objective is to create parameterized non-IID data partitions that better reflect the challenges observed in real-world settings. While these partitions may lead to lower accuracy under standard algorithms, the primary aim is not merely to degrade performance, or merely to induce difficulty through extreme or artificial splits, but to stress-test various state-of-the-art FL algorithms under diverse, controlled, and structured heterogeneity.

6.3.1 Analysis of Data Heterogeneity Under Realistic Partitioning Strategies

To further assess the impact and realism of our approach against various non-IID partitioning strategies in FL, we conduct a group of experiments using two distinct datasets, namely the CIFAR-10 and the FashionMNIST. We compare our approach against three widely referenced distribution approaches from state-of-the-art FL environment in Flower [138]: (1) exponential, (2) Dirichlet, and (3) Distribution-based. In our comparisons we quantify the heterogeneity and sparsity introduced by each method against the aforementioned benchmarks using three key metrics as follows:

1. **Entropy**: This measures the diversity of label classes within each client. A lower value indicates more skewed (i.e., less diverse) local datasets, representing higher non-IIDness. Real-world clients often tend to have naturally low entropy due to domain-specific data. Therefore, we use this metric essentially to demonstrate how localized or homogeneous each client's data is.

2. **Sparsity**: Defined as the number of classes missing from a client's dataset. High sparsity indicates that many classes are absent from most clients, reflecting realistic federated settings where clients rarely possess full class coverage. This is especially important when assessing whether a partitioning method reflects plausible real-world label silos.

3. **Jensen-Shannon (JS) Divergence**: JS divergence measures how dissimilar label distributions are between clients. Higher values indicate greater variability and uniqueness of data distributions across clients; a crucial property for evaluating the stress placed on FL algorithms under heterogeneous conditions.

We believe these metrics collectively provide a multifaceted view of intra-client label concentration (entropy), inter-client label exclusivity (sparsity), and global client diversity (JS divergence). Each method is applied over 1,000 clients, and the resulting label distributions are visualized in figures 6.1 and 6.2 with a sample of 20 clients from each strategy for visualization.

In the first set of experiments (Figure 6.1, Table 6.1), we utilize the CIFAR-10 dataset. The Exponential partitioning yields the highest average entropy (2.15) and the lowest sparsity (0.46), indicating that clients receive data from a broad range of classes, resulting in distributions close to IID. Visually, this is confirmed in the Exponential panel of figure 6.1, where most clients exhibit well-mixed label compositions. In contrast, Dirichlet and Distribution-based partitioning introduce more skew, with average entropy values dropping to approximately 1.56 and 1.58, and sparsity increasing to around 3. The visualizations reinforce this trend by displaying increasingly unbalanced label coverage across clients. However, our approach exhibits the most distinctive and realistic non-IID characteristics, with the lowest average entropy (1.26), highest sparsity (4.46), and highest JS divergence (0.64). This highlights that our method creates clients with strongly concentrated, yet diverse, class compositions.

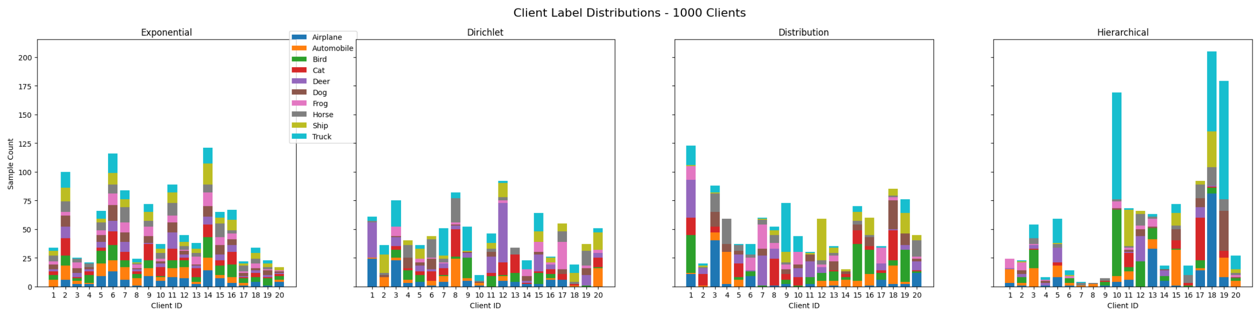


Figure 6.1 Label distribution histograms for 20 randomly sampled clients out of 1,000 in the CIFAR-10 dataset under four different partitioning strategies: Exponential, Dirichlet, Distribution-based, and Hierarchical Dirichlet (For single-level Dirichlet: α=0.5, for hierarchical Dirichlet: K=5, α=0.5, β=0.1).

Table 6.1 Comparison of Data Partitioning Methods on Entropy, Sparsity, and JS Divergence (For single-level Dirichlet: α=0.5, for hierarchical Dirichlet: K=5, α=0.5, β=0.1)

Method	**Avg. Entropy**	**Avg. Sparsity**	**Avg. JS Divergence**
Exponential	2.15	0.46	0.25
Dirichlet	1.56	3.00	0.58
Distribution	1.58	2.94	0.57
Hierarchical	1.26	4.46	0.64

A similar pattern is observed in the experiment utilizing the FashionMNIST dataset (Figure 6.2, Table 6.2). The Exponential baseline again yields the most IID-like configuration, with the highest entropy (2.18) and minimal sparsity (0.30). Moving to Dirichlet and Distribution approaches, entropy drops to around 1.57–1.58, and sparsity rises modestly (2.75–2.81). On the other hand, our approach achieves the lowest entropy (1.14), highest sparsity (4.64), and greatest JS divergence (0.65). The visual histogram confirms these insights, with clients under Hierarchical partitioning often dominated by a single or few label classes, yet differing strongly from one another.

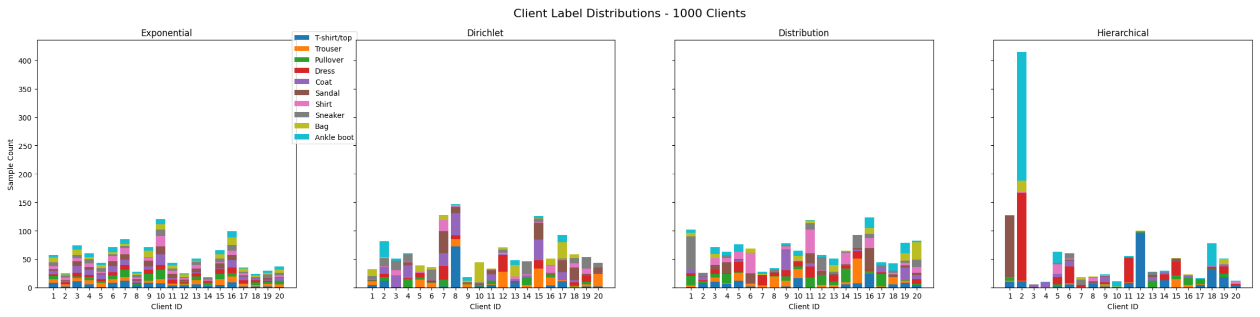


Figure 6.2 Label distribution histograms for 20 randomly sampled clients out of 1,000 in the FashionMNIST dataset under four different partitioning strategies: Exponential, Dirichlet, Distribution-based, and Hierarchical Dirichlet (For single-level Dirichlet: α=0.5, for hierarchical Dirichlet: K=10, α=0.5, β=0.1).

Table 6.2 Comparison of Data Partitioning Methods on FashionMNIST (For single-level Dirichlet: α=0.5, for hierarchical Dirichlet: K=10, α=0.5, β=0.1)

Method	**Avg. Entropy**	**Avg. Sparsity**	**Avg. JS Divergence**
Exponential	2.18	0.30	0.24
Dirichlet	1.57	2.81	0.56
Distribution	1.58	2.75	0.55
Hierarchical	1.14	4.64	0.65

To further ascertain our findings, we perform extended set of experiments with different parameters and environmental settings.

Table 6.3 Comparison of Data Partitioning Methods on Entropy, Sparsity, and JS Divergence (CIFAR-10, For single-level Dirichlet: α=0.1, for hierarchical Dirichlet: K=10, α=0.1, β=0.1)

Method	**Avg. Entropy**	**Avg. Sparsity**	**Avg. JS Div.**
Exponential	2.16	0.44	0.26
Dirichlet	1.56	3.00	0.57
Distribution	1.55	3.06	0.56
Hierarchical	1.07	5.27	0.68

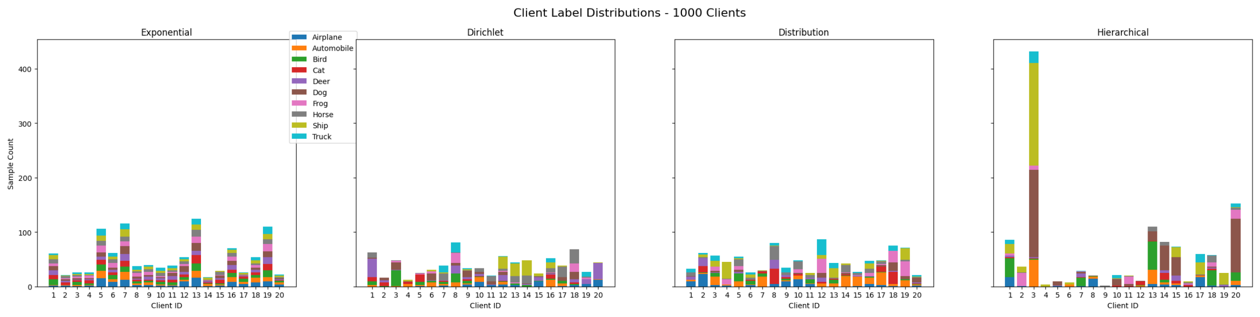


Figure 6.3 Label distribution histograms for 20 randomly sampled clients out of 1,000 in the CIFAR-10 dataset under four different partitioning strategies: Exponential, Dirichlet, Distribution-based, and Hierarchical Dirichlet (For single-level Dirichlet: α=0.1, for hierarchical Dirichlet: K=10, α=0.1, β=0.1).

In the set of experiments (Figure 6.3, Table 6.3), we evaluate the performance of various partitioning strategies on the CIFAR-10 dataset under strong non-IID conditions. The Exponential partitioning yields the highest average entropy (2.16) and the lowest sparsity (0.44), suggesting that clients receive a relatively broad mix of class labels, resulting in data distributions that are closer to IID. This indicates that while some class imbalance is introduced, the overall diversity at each client remains fairly high. In contrast, the Dirichlet and Distribution-based partitioning methods lead to lower average entropy values (1.56 and 1.55, respectively) and higher sparsity scores (around 3), signifying that clients are exposed to a more restricted set of classes, thereby increasing the degree of heterogeneity across the federation. Our proposed Hierarchical partitioning exhibits non-IID characteristics, with the lowest average entropy (1.07), the highest sparsity (5.27), and the highest JS divergence (0.68). This indicates that clients' datasets are heavily concentrated around fewer classes, with very limited class overlap between clients.

In another set of experiments (Figure 6.4, Table 6.4), we evaluate the behavior of different partitioning strategies on the FashionMNIST dataset. The Exponential partitioning achieves the highest average entropy (2.19) and the lowest sparsity (0.31), indicating that clients receive samples from a wide variety of classes, leading to distributions that are closer to IID. As expected, this suggests minimal heterogeneity among clients. In contrast, the Dirichlet and Distribution-based partitioning methods reduce the average entropy to around 1.58 and 1.59, respectively, while increasing sparsity to approximately 2.8, suggesting that client datasets become more concentrated around a smaller number of classes. Our proposed Hierarchical partitioning method results in the most extreme non-IID setting, characterized by a drastically low average entropy (0.14), very high sparsity (8.56), and the highest JS divergence (0.75) among all methods. This configuration creates highly specialized clients where most clients have data from only one or very few classes, with minimal overlap between client datasets.

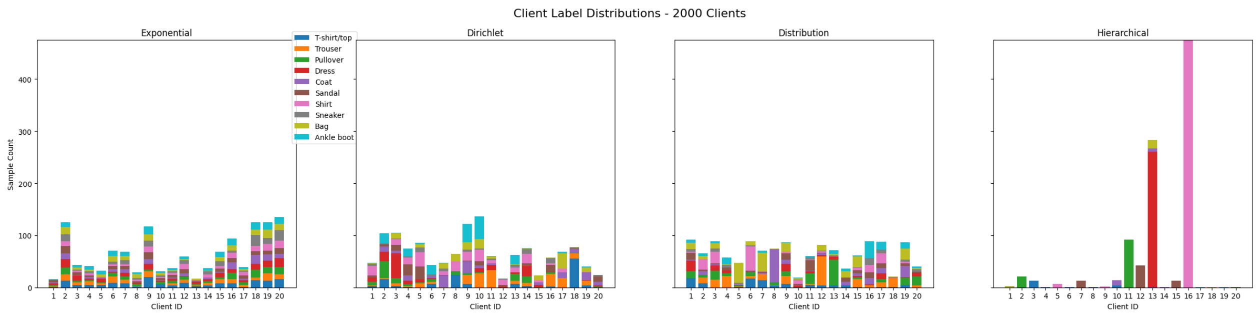


Figure 6.4 Label distribution histograms for 20 randomly sampled clients out of 1,000 in the FashionMNIST dataset under four different partitioning strategies: Exponential, Dirichlet, Distribution-based, and Hierarchical Dirichlet (For single-level Dirichlet: α=0.1, for hierarchical Dirichlet: K=5, α=0.1, β=0.01).

Table 6.4 Comparison of Data Partitioning Methods on Entropy, Sparsity, and JS Divergence (FashionMNIST, For single-level Dirichlet: α=0.1, for hierarchical Dirichlet: K=5, α=0.1, β=0.01)

Method	**Avg. Entropy**	**Avg. Sparsity**	**Avg. JS Div.**
Exponential	2.19	0.31	0.23
Dirichlet	1.58	2.79	0.56
Distribution	1.59	2.80	0.56
Hierarchical	0.14	8.56	0.75

These results collectively highlight that the ability to control both intra-cluster label allocation (via β) and inter-cluster client allocation (via α) results in a flexible way, allows us to create diverse, sparse and divergent distributions. In fact, the metrics used provide a rigorous lens into this behavior. Entropy reflects per-client label diversity, helping assess the local learning complexity. Sparsity captures label absence per client, simulating under-represented categories or silos. JS divergence quantifies the dissimilarity across clients, enabling us to understand how isolated or varied each client is relative to others. These experiments also validate the scalability, expressiveness, and practical utility of our proposed approach using different datasets. It maintains its distinguishing characteristics even at the 1,000-client scale offering a tunable benchmark for evaluating FL methods in heterogeneous environments.

6.3.2 Federated Training Accuracy under Heterogeneous Partitioning Strategies

To further assess the implications of different data partitioning strategies on learning performance, we simulate federated training with 1,000 clients under partial participation for 300 rounds, sampling between 10 and 20 clients per round. We run the experiment under two differnt cluster settings; mainly $K = 5$ and $K = 10$; using our proposed Hierarchical Dirichlet method alongside Exponential, Dirichlet, and Linear partitioning baselines to assess the implications of different data partitioning strategies on learning performance. We benchmark three optimization approaches; ClusterFL [139], FedBN [140], and FedL2P [141]; which employ varying personalization and adaptation mechanisms for non-IID data, demonstrating that our approach offers a challenging yet learnable environment representative of realistic federated settings. We run these experiments with $\alpha = 0.1$ and $\beta = 0.01$ for our approach, and with $\alpha = 0.5$ for the single-level Dirichlet approach using the CIFAR-10 dataset.

It is important to highlight that, in this set of experiments, our goal is to demonstrate how specialized non-IID optimization algorithms respond to the structured heterogeneity introduced by our approach, offering insights into their robustness and adaptability. We do not consider model accuracy degradation as sufficient validation of our partitioning strategy. Instead, our primary objective is to replicate realistic forms of data heterogeneity found in federated deployments; such as regional label imbalance, feature subpopulation divergence, and client-specific specialization, and highlight how state-of-the-art FL optimization techniques, designed to mitigate the effects of non-IID react to our distribution approach. Accordingly, in this context, accuracy is only one observable outcome and must be interpreted alongside structural and statistical measures, which we examine empirically in section 6.3.1.

Under FedAvg (Figure 6.5), the hierarchical approach converges around 32%, while the baselines reach approximately 40%–41%, indicating that standard aggregation struggles under structured heterogeneity. ClusterFL (Figure 6.6) improves this gap slightly, with the hierarchical method reaching 34% as it benefits from client clustering. FedBN (Figure 6.7) achieves high accuracies (57%–60%) for the baselines, but the hierarchical approach remains challenging at 43%, suggesting that local normalization alone cannot fully address feature-level skew. FedL2P (Figure 6.8) performs best overall, achieving up to 56% with baseline partitioners, while the hierarchical method reaches 42%, showing that even prototype-based personalization faces difficulty under structured non-IID partitions. These consistent trends across algorithms validate the realism of our proposed method, which remains learnable yet exposes generalization limitations in existing FL strategies.

To assess the behavior of the partitioning strategies under a reduced number of clusters, we evaluate the test accuracy across 300 communication rounds with 5 clusters and 20 clients

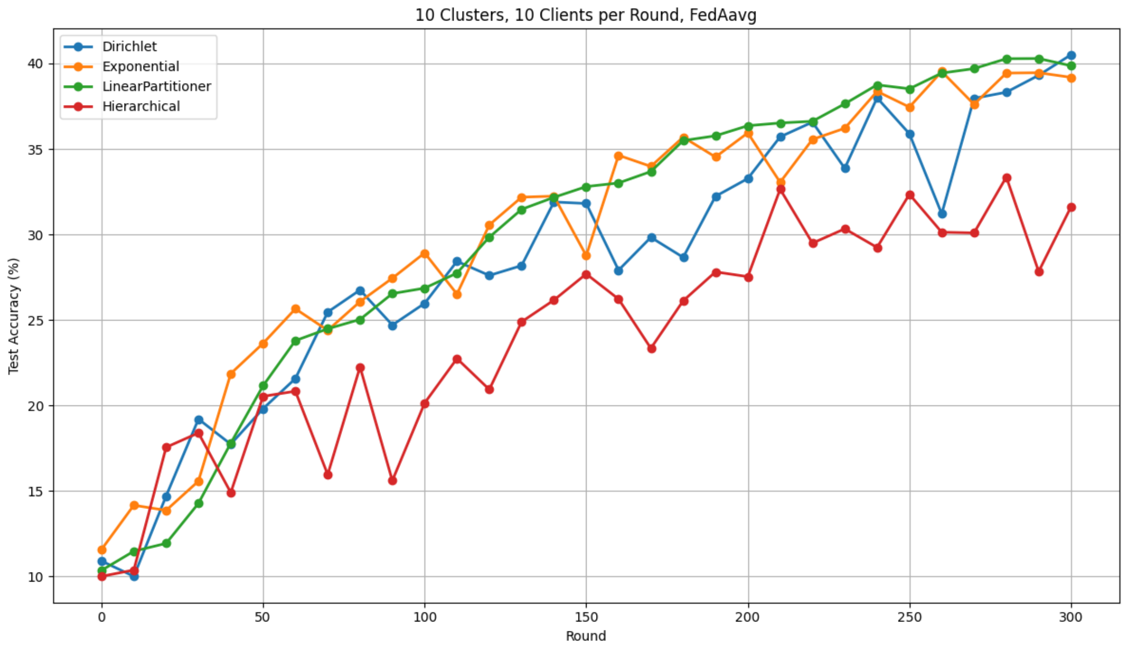


Figure 6.5 Test accuracy comparison using FedAvg for 10 clusters, 10 clients per round

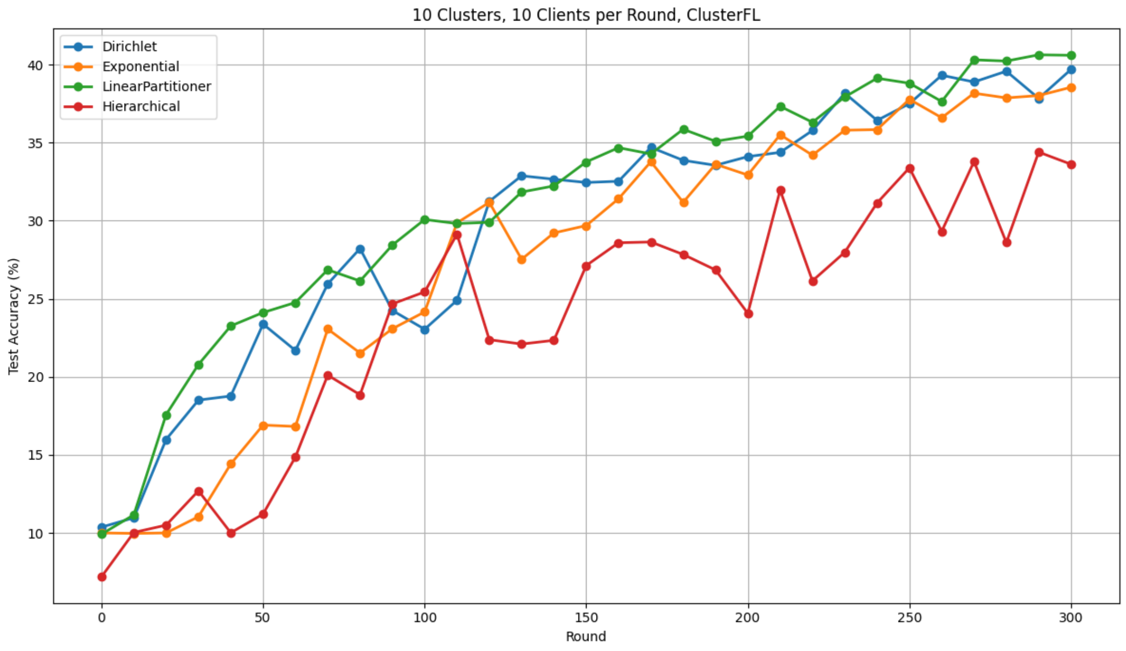


Figure 6.6 Test accuracy comparison using ClusterFL for 10 clusters, 10 clients per round

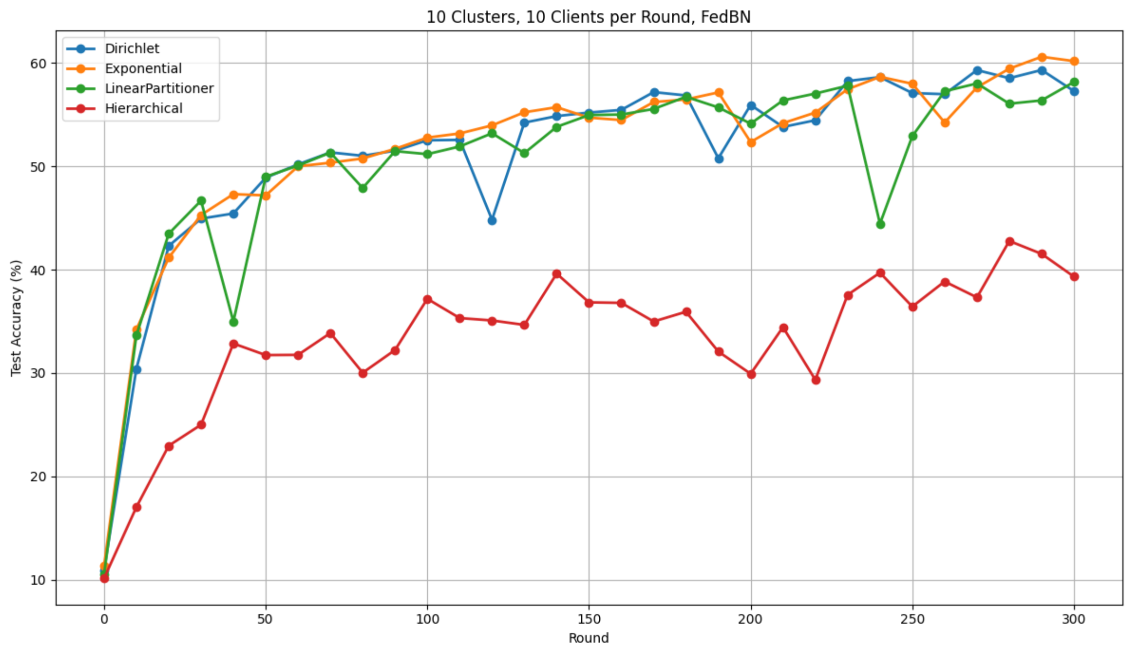


Figure 6.7 Test accuracy comparison using FedBN for 10 clusters, 10 clients per round

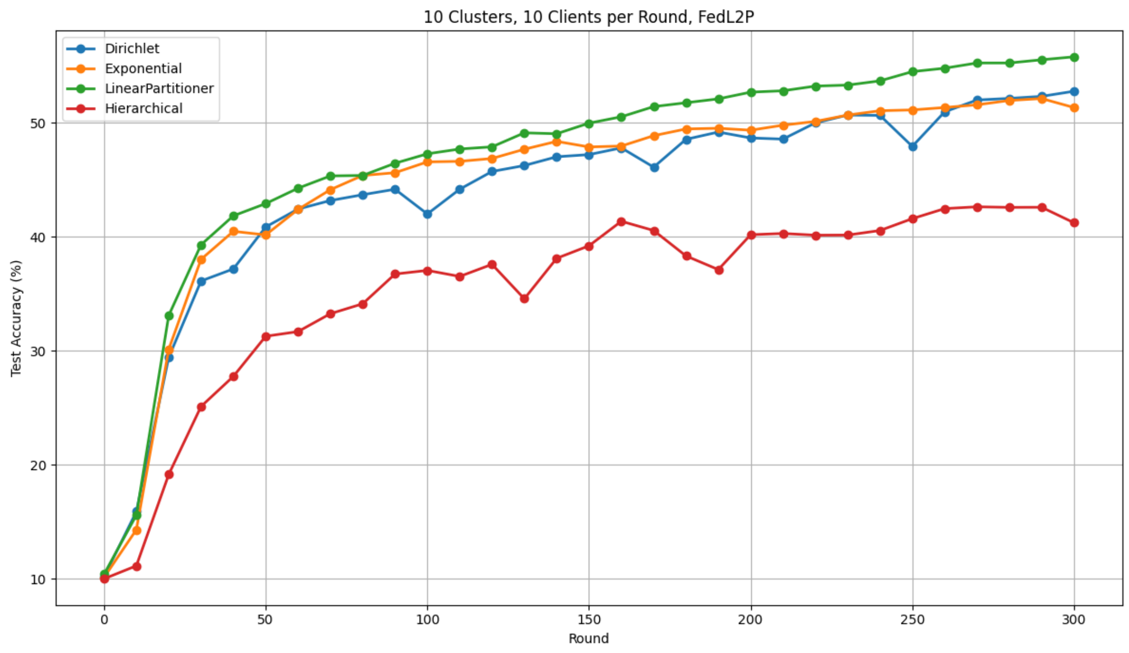


Figure 6.8 Test accuracy comparison using FedL2P for 10 clusters, 10 clients per round

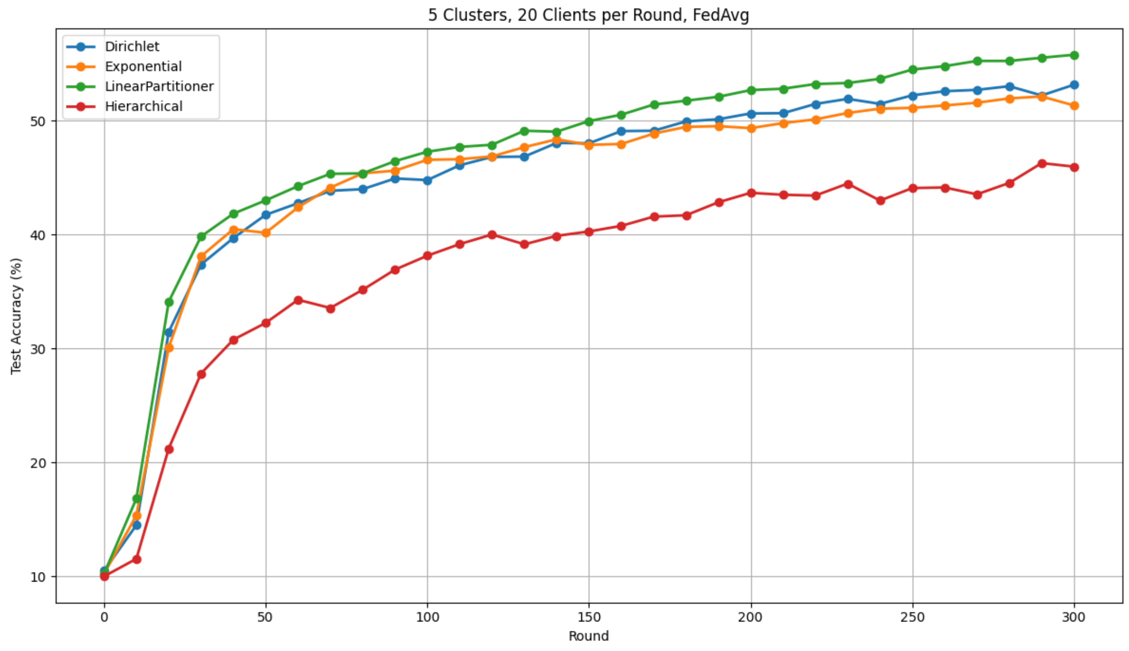


Figure 6.9 Test accuracy comparison using FedAvg for 5 clusters, 20 clients per round

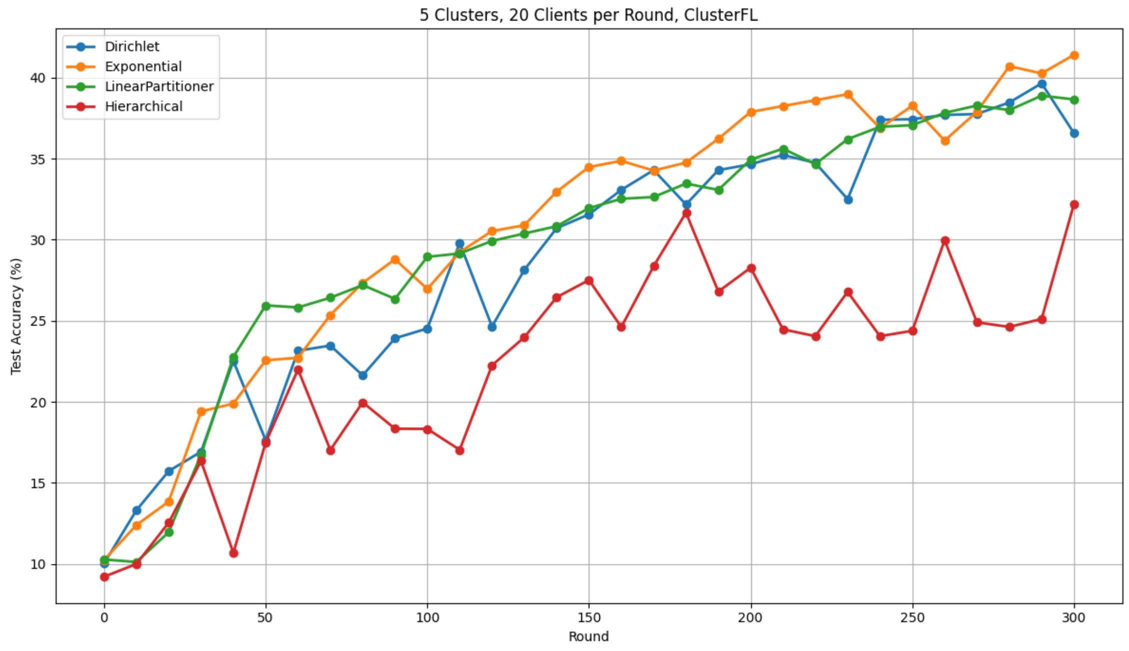


Figure 6.10 Test accuracy comparison using ClusterFL for 5 clusters, 20 clients per round

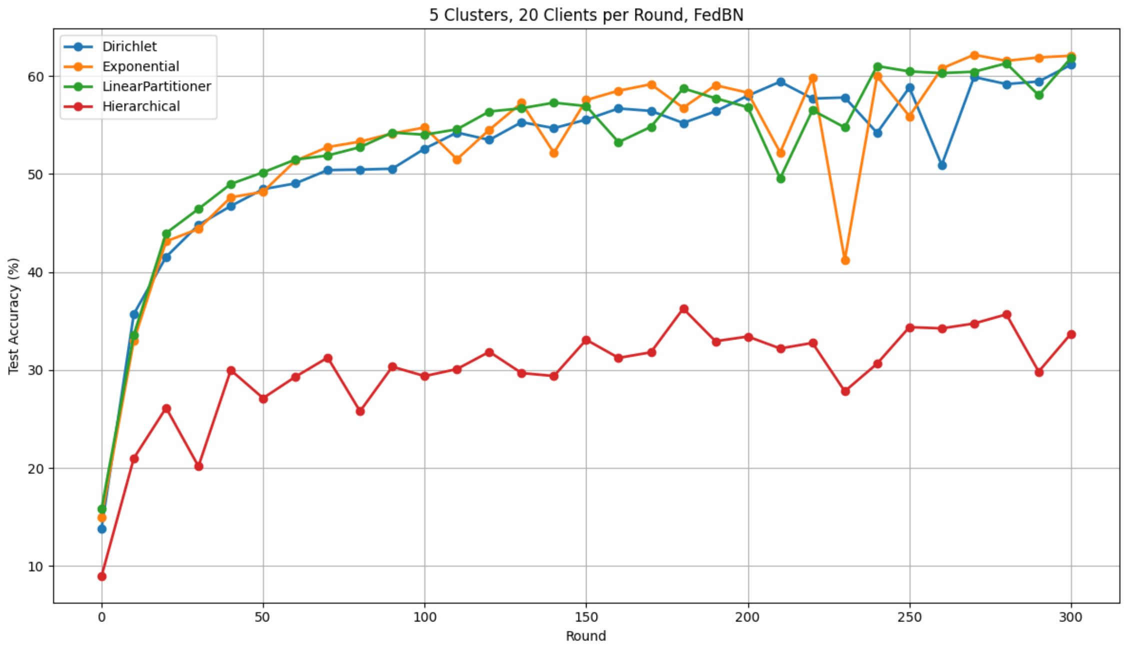


Figure 6.11 Test accuracy comparison using FedBN for 5 clusters, 20 clients per round

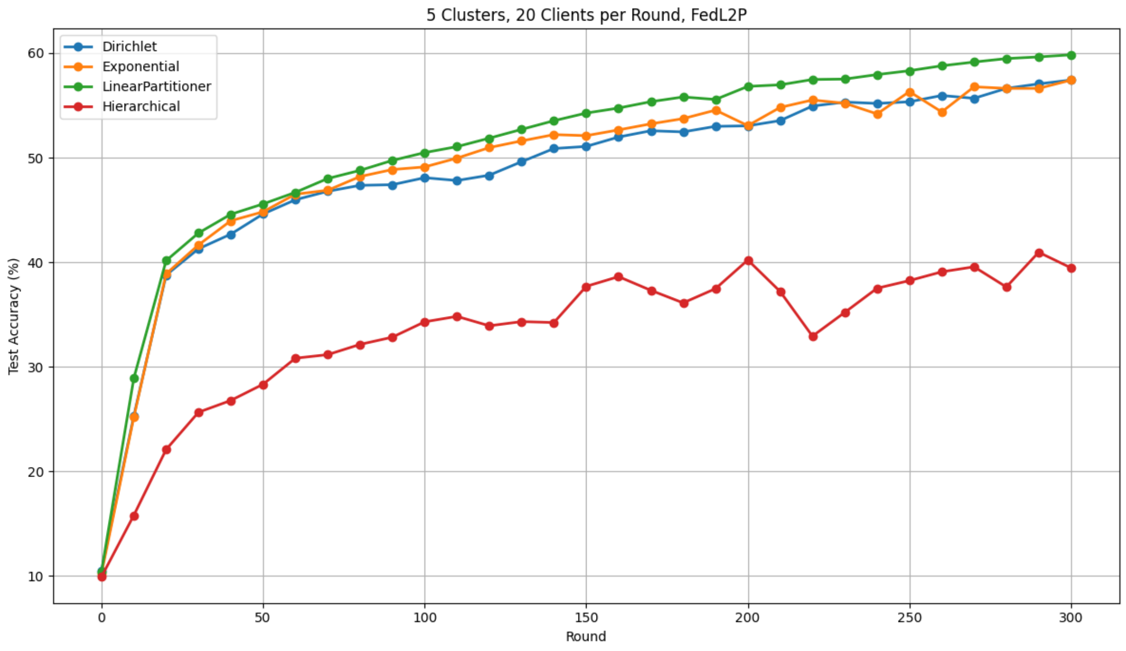


Figure 6.12 Test accuracy comparison using FedL2P for 5 clusters, 20 clients per round

per round. As shown in Figures 6.9, 6.10, 6.11, and 6.12, the hierarchical Dirichlet partitioning continues to present the most significant learning challenge compared to Dirichlet, Exponential, and Linear baselines. In the case of FedAvg (Figure 6.9), Linear reaches the highest final accuracy at approximately 56%, followed by Dirichlet and Exponential at around 53%–54%, while the hierarchical approach stabilizes at roughly 46%. This gap illustrates that reducing the number of clusters amplifies the label imbalance within clusters, which increases intra-cluster diversity and hinders model generalization. With ClusterFL (Figure 6.10), the Exponential strategy slightly outperforms others, converging near 42%, while the hierarchical setting achieves approximately 33%. Although ClusterFL leverages structural similarity, it remains sensitive to the deeper statistical misalignments embedded in the hierarchical partitions. FedBN (Figure 6.11) continues to perform strongly for all baseline partitioners, reaching around 60% accuracy, whereas the hierarchical method only attains approximately 35%. This substantial performance gap indicates that feature normalization alone is insufficient when both inter-cluster and intra-client disparities are present. Finally, FedL2P (Figure 6.12) shows consistent superiority in personalization, with baseline partitioners converging above 57%, and the hierarchical variant improving slightly over previous settings to around 40%. Despite the more aggressive personalization, this result still reflects the difficulty posed by our structured data generation. Collectively, these findings confirm that a lower number of clusters intensifies the heterogeneity challenge and further differentiates the realism of our method from artificially balanced baselines.

6.4 Conclusion

In this chapter, we proposed a Hierarchical Dirichlet partitioning method designed to enhance the realism of non-IID distributions in FL settings. Unlike single level Dirichlet distribution, our approach introduces a tunable and interpretable multi-level control structure: one parameter (α) governs the allocation of clients to semantic clusters, while another (β) controls label skew within each cluster. This enables us to synthesize federated scenarios where some clients may have broad label diversity, others narrow, and inter-client overlap is variable; all properties that are observed in real federated settings classification. This fine-grained controllability makes our method uniquely suited for evaluating federated algorithms, particularly in areas such as personalization, and fairness to data shifts.

CHAPTER 7 CONCLUSION & FUTURE WORKS

7.1 Summary of Works

In this thesis, we proposed novel and modular FL frameworks that advance fairness, decentralization, and economic incentive compatibility through a set of interrelated technical and game-theoretic contributions. The primary motivation of our work was to address the limitations of traditional FL pipelines in handling heterogeneous data distributions, unequal system resources, and unfair client selection practices. Our approach comprises four tightly coupled contributions that introduce a fair and incentive-driven FL architectures that integrates coalition formation, trust quantification, incentive allocation, and realistic data partitioning. Our solutions were designed and validated through simulation-based experiments, revealing consistent improvements in fairness, and robustness compared to state-of-the-art FL methodologies.

Specifically, we developed a novel client-to-client trust framework that combines both subjective (recommendation-based) and objective (monitoring-based) trust indicators to filter out untrustworthy or selfish clients in FL. Experimental results reveal that this framework improves model robustness and convergence stability when compared to FL schemes that ignore behavioral trust or rely solely on server-driven metrics. Secondly, we introduced a decentralized coalition formation protocol based on a Stackelberg game between leader and follower clients. The leader clients autonomously select their followers, creating collaborative coalitions that abstract data and system heterogeneity from the central aggregator. The proposed solution is shown to outperform traditional selection schemes in terms of coalition stability and client inclusivity, particularly when resource imbalance exists. Thirdly, we designed an internal reputation-based federated marketplace where task owners, network owners, and clients interact via economically grounded auctions. In this marketplace, clients are rewarded according to their contributions, while network owners gain reputation and fees according to their ability to fulfill task owner's requirements. Finally, we introduced a feature-aware, two-level Dirichlet-based partitioning model that emulates realistic non-IID distributions in FL benchmarks. Unlike traditional Dirichlet approaches, our method builds on intra-cluster and inter-cluster diversity, enabling fine-grained control over client distributions. Experiments show that our partitioning method introduces measurable heterogeneity and challenge, validating its utility as a benchmark for robustness-oriented FL evaluations.

Accordingly, the following points summarize the main contributions of this thesis:

- We proposed a trust-aware client selection mechanism that leverages both objective (resource and behavior monitoring) and subjective (recommendation-based) metrics to compute credibility scores, thus promoting honest and stable client participation in FL, by giving clients the autonomy to select the coalitions they want to be part of.

- We put forward a novel client-centric coalition formation framework based on Stackelberg games in FL. Our model enables resourceful clients to form strategic alliances with less resourceful ones, providing a self-organizing alternative to server-driven selection, and demonstrating convergence to stable coalitions.

- We proposed an FL marketplace that combines reverse auctions with reputation-based incentives to ensure fairness, incentive compatibility, and economic sustainability. The framework encourages continued participation for clients by rewarding reliable contributors, fostering a self-regulating and trustworthy learning ecosystem.

- We introduced a feature-aware, hierarchical two-level Dirichlet data partitioning framework that enables tunable and realistic non-IID data generation, surpassing traditional benchmark methods in replicating different levels of data heterogeneity.

The first contribution is proposed to answer our first research objective (Objective 1), which focused on enabling a trust-aware client selection framework that improves model reliability by filtering out unreliable or selfish clients. The second contribution is proposed to answer our second research objective (Objective 2), which aimed to develop a decentralized community formation mechanism through a Stackelberg-game-based leader-follower model, enhancing fairness and inclusivity in federated environments. The third contribution is proposed to answer our third research objective (Objective 3), which sought to establish an incentive-compatible federated marketplace that aligns clients' financial motivations with system-level goals through auction-based mechanisms and internal reputation systems. Finally, the fourth contribution is proposed to address our fourth research objective (Objective 4), which targeted the creation of heterogeneous data simulation framework using a hierarchical two-level Dirichlet distribution to emulate different levels of non-IID federated settings.

7.2 Limitations and Future Research Directions

While this thesis proposes robust frameworks for trust-aware client selection, coalition formation, federated marketplaces, and realistic data simulation, several avenues remain open for further enhancement.

Hereafter, we outline several directions aimed at inspiring future research and exploration:

- **Adaptivity to Evolving Threats:** While the proposed trust and reputation mechanisms effectively identify and filter dishonest or selfish clients, adversaries may continuously adapt their strategies to evade detection. Future work could focus on developing adaptive trust recalibration frameworks that evolve alongside emerging threat patterns. Possible avenues of achieving this could leverage online learning, adversarial modeling, or reinforcement learning techniques to dynamically update trust weights based on behavioral deviations, historical performance, and contextual feedback.

- **Multi-Stage Stackelberg Games:** Future work could extend the proposed Stackelberg-based coalition framework into multi-stage hierarchical structures involving multiple leaders and followers. In such settings, leaders may compete or collaborate across tiers, reflecting different topologies such as regional hubs or federations of edge clusters. Studying these cross-coalition dynamics would provide deeper insights into equilibrium behaviors, stability, and efficiency in large-scale, decentralized federated systems.

- **Interoperable Federated Marketplaces:** The design of communication and exchange protocols that allow heterogeneous federated marketplaces can be another promising direction. Each marketplace may be governed by distinct policies, currency units, or reputation systems, yet designed to interoperate seamlessly. Such interoperability would enable cross-market collaboration, value transfer, and model exchange, functioning analogously to cross-chain bridges in blockchain ecosystems and paving the way for a global FL economy.

- **Reputation Tokenization and Data Economy Integration:** Future research could explore the tokenization of reputation as a transferable digital asset. By enabling clients to carry their verified credibility across multiple tasks or marketplaces, this approach could create a global reputation token and give rise to broader 'Federated Learning Reputation Economy', in which reputation becomes both an indicator of reliability and a tradable economic entity, useable across interoperable federated marketplaces.

- **Decentralized Peer-to-Peer Marketplaces:** Another promising direction is the design of fully decentralized federated marketplaces governed through blockchain and smart contracts. This would eliminate reliance on a central network owner, enhancing transparency, trust, and autonomy among participating agents.

- **Federated Governance and Policy Simulation:** Another promising avenue involves the development of governance frameworks where participants collectively propose, vote on, and refine reputation, trust, and incentive policies. Such participatory governance would create 'federated democracy', and empower stakeholders to adapt system rules dynamically, promoting transparency, accountability, and fairness across the evolving federated ecosystem.

- **Data Simulation Extensions:** While the hierarchical two-level Dirichlet framework significantly enhances realism, future work could integrate temporal data dynamics and hierarchical class relationships to simulate even more complex real-world data distributions.

The limitations and future directions outlined above represent opportunities to further strengthen the robustness, scalability, and real-world applicability of the solutions proposed in this thesis. They are natural continuations of the flexible foundations established herein, offering a rich landscape for future innovation.

7.3 Thesis Conclusion

In conclusion, this thesis presented a comprehensive framework to advance FL environments by improving trust, fairness, and adaptability across federated networks. Through the integration of trust-aware client selection, coalition-based participation models, incentive-compatible marketplaces, and enhanced non-IID data simulations, we addressed key technical and economic challenges in the realm of FL. The proposed methodologies not only enhance system robustness but also open new pathways for building sustainable, privacy-preserving AI ecosystems. Across all experiments in chapters 3, 4, and 5, our approaches consistently outperformed baselines and state-of-the-art methods under the evaluated datasets and experimental settings, while maintaining high model accuracy under non-IID settings. Furthermore, in chapter 6, our approach was able to create non-IID distributions with lower entropy and higher sparsity scores compared to other data-distribution techniques. We anticipate that the foundations laid in this work provide a pathway for future innovation, shaping future research and practical deployments across diverse application domains.

REFERENCES

[1] 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.

[2] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtarik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," in *NIPS Workshop on Private Multi-Party Machine Learning*, 2016. [Online]. Available: https://arxiv.org/abs/1610.05492

[3] S. Vahidian, M. Morafah, C. Chen, M. Shah, and B. Lin, "Rethinking data heterogeneity in federated learning: Introducing a new notion and standard benchmarks," *IEEE Transactions on Artificial Intelligence*, vol. 5, no. 3, pp. 1386–1397, 2024.

[4] Y. Zhang, W. Zhang, L. Pu, T. Lin, and J. Yan, "To distill or not to distill: Toward fast, accurate, and communication-efficient federated distillation learning," *IEEE Internet of Things Journal*, vol. 11, no. 6, pp. 10 040–10 053, 2024.

[5] M. Ali, F. Naeem, M. Tariq, and G. Kaddoum, "Federated learning for privacy preservation in smart healthcare systems: A comprehensive survey," *IEEE Journal of Biomedical and Health Informatics*, vol. 27, no. 2, pp. 778–789, 2023.

[6] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *arXiv preprint arXiv:1912.04977*, 2019.

[7] O. A. Wahab, A. Mourad, H. Otrok, and T. Taleb, "Federated machine learning: Survey, multi-level classification, desirable criteria and future directions in communication and networking systems," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 1342–1397, 2021.

[8] K. Daly, H. Eichner, P. Kairouz, H. B. McMahan, D. Ramage, and Z. Xu, "Federated learning in practice: Reflections and projections," in *2024 IEEE 6th International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications (TPS-ISA)*, 2024, pp. 148–156.

[9] S. Arisdakessian, O. A. Wahab, A. Mourad, and H. Otrok, “Towards instant clustering approach for federated learning client selection,” in *2023 International Conference on Computing, Networking and Communications (ICNC)*, 2023, pp. 409–413.

[10] S. Arisdakessian, O. Wehbi, O. A. Wahab, A. Mourad, and H. Otrok, “Towards vox populi in federated learning: A fair and inclusive client selection framework,” *IEEE Transactions on Artificial Intelligence*, pp. 1–15, 2025.

[11] S. Arisdakessian, O. Wehbi, O. A. Wahab, A. Mourad, H. Otrok, and M. Guizani, “A two-level dirichlet framework for heterogeneous federated network,” *IEEE Transactions on Network Science and Engineering*, pp. 1–18, 2025.

[12] S. Abdulrahman, H. Tout, H. Ould-Slimane, A. Mourad, C. Talhi, and M. Guizani, “A survey on federated learning: The journey from centralized to distributed on-site learning and beyond,” *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5476–5497, 2021.

[13] Q. Li, Y. Diao, Q. Chen, and B. He, “Federated learning on non-iid data silos: An experimental study,” in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, 2022, pp. 965–978.

[14] H. Zhu, J. Xu, S. Liu, and Y. Jin, “Federated learning on non-iid data: A survey,” *Neurocomputing*, vol. 465, pp. 371–390, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231221013254

[15] L. Zhang, L. Shen, L. Ding, D. Tao, and L.-Y. Duan, “Fine-tuning global model via data-free knowledge distillation for non-iid federated learning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 10 174–10 183.

[16] L. Zhang, Y. Luo, Y. Bai, B. Du, and L.-Y. Duan, “Federated learning for non-iid data via unified feature learning and optimization objective alignment,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 4420–4428.

[17] M. Luo, F. Chen, D. Hu, Y. Zhang, J. Liang, and J. Feng, “No fear of heterogeneity: Classifier calibration for federated learning with non-iid data,” in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021,

pp. 5972–5984. [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/2f2b265625d76a6704b08093c652fd79-Paper.pdf

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

[19] M. Arafeh, A. Hammoud, H. Otrok, A. Mourad, C. Talhi, and Z. Dziong, "Independent and identically distributed (iid) data assessment in federated learning," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, 2022, pp. 293–298.

[20] L. Qu, Y. Zhou, P. P. Liang, Y. Xia, F. Wang, E. Adeli, L. Fei-Fei, and D. Rubin, "Rethinking architecture design for tackling data heterogeneity in federated learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 061–10 071.

[21] T. Yoon, S. Shin, S. J. Hwang, and E. Yang, "Fedmix: Approximation of mixup under mean augmented federated learning," in *International Conference on Learning Representations (ICLR)*, 2021. [Online]. Available: https://arxiv.org/abs/2107.00233

[22] T. Li, S. Hu, A. Beirami, and V. Smith, "Ditto: Fair and robust federated learning through personalization," in *International conference on machine learning.* PMLR, 2021, pp. 6357–6368.

[23] H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," 2019. [Online]. Available: https://arxiv.org/abs/1909.06335

[24] K. Hsieh, A. Phanishayee, O. Mutlu, and P. B. Gibbons, "The non-iid data quagmire of decentralized machine learning," in *Proceedings of the 37th International Conference on Machine Learning*, ser. ICML'20. JMLR.org, 2020.

[25] F. Sattler, K.-R. Müller, and W. Samek, "Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 8, pp. 3710–3722, 2021.

[26] H. Chen, A. Frikha, D. Krompass, J. Gu, and V. Tresp, "Fraug: Tackling federated learning with non-iid features via representation augmentation," in *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 4826–4836.

[27] Z. Yang, Y. Zhang, Y. Zheng, X. Tian, H. Peng, T. Liu, and B. Han, "Fedfed: Feature distillation against data heterogeneity in federated learning," in *Advances*

in Neural Information Processing Systems, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36. Curran Associates, Inc., 2023, pp. 60 397–60 428. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/file/bdcdf38389d7fcefc73c4c3720217155-Paper-Conference.pdf

[28] X. Wu, H. Huang, Y. Ding, H. Wang, Y. Wang, and Q. Xu, "Fednp: Towards non-iid federated learning via federated neural propagation," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 9, pp. 10 399–10 407, Jun. 2023. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/26237

[29] Z. Tang, Y. Zhang, S. Shi, X. He, B. Han, and X. Chu, "Virtual homogeneity learning: Defending against data heterogeneity in federated learning," in *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds., vol. 162. PMLR, 17–23 Jul 2022, pp. 21 111–21 132. [Online]. Available: https://proceedings.mlr.press/v162/tang22d.html

[30] M. Luo, F. Chen, D. Hu, Y. Zhang, J. Liang, and J. Feng, "No fear of heterogeneity: Classifier calibration for federated learning with non-iid data," 2021. [Online]. Available: https://arxiv.org/abs/2106.05001

[31] W. Chen and Q. Qiu, "Extra clients at no extra cost: Overcome data heterogeneity in federated learning with filter decomposition," 2025. [Online]. Available: https://arxiv.org/abs/2503.08652

[32] S. Chen and B. Li, "Towards optimal multi-modal federated learning on non-iid data with hierarchical gradient blending," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*. IEEE Press, 2022, p. 1469–1478. [Online]. Available: https://doi.org/10.1109/INFOCOM48880.2022.9796724

[33] C. T. Dinh, N. H. Tran, and T. D. Nguyen, "Personalized federated learning with moreau envelopes," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS '20. Red Hook, NY, USA: Curran Associates Inc., 2020.

[34] E. G. Valente Neto, S. A. Peixoto, V. R. Q. Leithardt, J. F. De Paz, and J. C. S. Dos Anjos, "Adding data quality to federated learning performance improvement," *IEEE Access*, pp. 1–1, 2025.

[35] M. A. Hoefler, T. Mazouka, K. Mueller, and W. Samek, “Boosting federated learning with diffusion models for non-iid and imbalanced data,” in *2024 IEEE International Conference on Big Data (BigData)*, 2024, pp. 7790–7799.

[36] H. Sun, H. Zhao, L. Xu, R. Zhang, W. Zhang, W. Jiang, H. Guan, and B. Zhang, “Diffusion generation-based federated learning for non-iid defect recognition,” *IEEE Transactions on Computational Social Systems*, pp. 1–14, 2025.

[37] Z. Li, Y. Sun, J. Shao, Y. Mao, J. H. Wang, and J. Zhang, “Feature matching data synthesis for non-iid federated learning,” *IEEE Transactions on Mobile Computing*, vol. 23, no. 10, pp. 9352–9367, 2024.

[38] P. Zhao, S. Guo, Y. Li, S. Yang, and X. Ren, “Fedgen: Personalized federated learning with data generation for enhanced model customization and class imbalance,” *Future Generation Computer Systems*, vol. 164, p. 107595, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X24005594

[39] Y. Cai, W. Xi, Y. Shen, C. Sun, S. Wang, W. Gong, and J. Zhao, “Individualized data generation in personalized federated learning,” *IEEE Transactions on Mobile Computing*, vol. 24, no. 7, pp. 6628–6642, 2025.

[40] Y. Fraboni, R. Vidal, and M. Lorenzi, “Free-rider attacks on model aggregation in federated learning,” in *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, A. Banerjee and K. Fukumizu, Eds., vol. 130. PMLR, 13–15 Apr 2021, pp. 1846–1854. [Online]. Available: https://proceedings.mlr.press/v130/fraboni21a.html

[41] N. Zhang, Q. Ma, and X. Chen, “Enabling long-term cooperation in cross-silo federated learning: A repeated game perspective,” *IEEE Transactions on Mobile Computing*, pp. 1–1, 2022.

[42] J. Wang, “Pass: Parameters audit-based secure and fair federated learning scheme against free rider,” 2022. [Online]. Available: https://arxiv.org/abs/2207.07292

[43] S. P. Karimireddy, W. Guo, and M. I. Jordan, “Mechanisms that incentivize data sharing in federated learning,” 2022. [Online]. Available: https://arxiv.org/abs/2207.04557

[44] F. Famá, C. Kalalas, S. Lagen, and P. Dini, “Measuring data similarity for efficient federated learning: A feasibility study,” 2024.

[45] C. Briggs, Z. Fan, and P. Andras, “Federated learning with hierarchical clustering of local updates to improve training on non-iid data,” in *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–9.

[46] L. Yu, W. Nie, L. Xin, and M. Guo, “Clustered federated learning based on data distribution,” in *Proceedings of the 3rd International Conference on Advanced Information Science and System*, ser. AISS ’21. New York, NY, USA: Association for Computing Machinery, 2022. [Online]. Available: https://doi.org/10.1145/3503047.3503102

[47] B. Youssef, D. Jabir, and K. Abdellatif, “Impact of selfish nodes on federated learning performances,” in *2022 9th International Conference on Wireless Networks and Mobile Communications (WINCOM)*, 2022, pp. 1–6.

[48] J. Lin, M. Du, and J. Liu, “Free-riders in federated learning: Attacks and defenses,” *CoRR*, vol. abs/1911.12560, 2019. [Online]. Available: http://arxiv.org/abs/1911.12560

[49] B. Wang, H. Li, X. Liu, and Y. Guo, “Frad: Free-rider attacks detection mechanism for federated learning in aiot,” *IEEE Internet of Things Journal*, vol. 11, no. 3, pp. 4377–4388, 2024.

[50] S. AbdulRahman, H. Tout, A. Mourad, and C. Talhi, “Fedmccs: multicriteria client selection model for optimal iot federated learning,” *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4723–4735, 2020.

[51] G. Rjoub, O. A. Wahab, J. Bentahar, and A. Bataineh, “A trust and energy-aware double deep reinforcement learning scheduling strategy for federated learning on iot devices,” in *International Conference on Service-Oriented Computing.* Springer, 2020, pp. 319–333.

[52] M. Chen, H. V. Poor, W. Saad, and S. Cui, “Convergence time optimization for federated learning over wireless networks,” *IEEE Transactions on Wireless Communications*, vol. 20, no. 4, pp. 2457–2471, 2020.

[53] M. Chahoud, S. Otoum, and A. Mourad, “On the feasibility of federated learning towards on-demand client deployment at the edge,” *Information Processing Management*, vol. 60, no. 1, p. 103150, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306457322002515

[54] W. Zhang, X. Wang, P. Zhou, W. Wu, and X. Zhang, "Client selection for federated learning with non-iid data in mobile edge computing," *IEEE Access*, vol. 9, pp. 24 462–24 474, 2021.

[55] H. Huang, W. Shi, Y. Feng, C. Niu, G. Cheng, J. Huang, and Z. Liu, "Active client selection for clustered federated learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 11, pp. 16 424–16 438, 2024.

[56] S. Guo, H. Wang, S. Lin, Z. Kou, and X. Geng, "Addressing skewed heterogeneity via federated prototype rectification with personalization," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–13, 2024.

[57] Y. Zhan, J. Zhang, Z. Hong, L. Wu, P. Li, and S. Guo, "A survey of incentive mechanism design for federated learning," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 2, pp. 1035–1044, 2021.

[58] A. Ali, I. Ilahi, A. Qayyum, I. Mohammed, A. Al-Fuqaha, and J. Qadir, "A systematic review of federated learning incentive mechanisms and associated security challenges," *Comput. Sci. Rev.*, vol. 50, no. C, Nov. 2023. [Online]. Available: https://doi.org/10.1016/j.cosrev.2023.100593

[59] K. Wang, Y. Ma, M. B. Mashhadi, C. H. Foh, R. Tafazolli, and Z. Ding, "Convergence acceleration in wireless federated learning: A stackelberg game approach," *IEEE Transactions on Vehicular Technology*, pp. 1–15, 2024.

[60] L. U. Khan, S. R. Pandey, N. H. Tran, W. Saad, Z. Han, M. N. H. Nguyen, and C. S. Hong, "Federated learning for edge networks: Resource optimization and incentive mechanism," *IEEE Communications Magazine*, vol. 58, no. 10, pp. 88–93, 2020.

[61] Y. Chen, Z. Li, B. Yang, K. Nai, and K. Li, "A stackelberg game approach to multiple resources allocation and pricing in mobile edge computing," *Future Generation Computer Systems*, vol. 108, pp. 273–287, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X19311653

[62] G. Xiao, M. Xiao, G. Gao, S. Zhang, H. Zhao, and X. Zou, "Incentive mechanism design for federated learning: A two-stage stackelberg game approach," in *2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS)*, 2020, pp. 148–155.

[63] Y. Zhan, P. Li, Z. Qu, D. Zeng, and S. Guo, “A learning-based incentive mechanism for federated learning,” *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6360–6368, 2020.

[64] Y. Chen, H. Zhou, T. Li, J. Li, and H. Zhou, “Multifactor incentive mechanism for federated learning in iot: A stackelberg game approach,” *IEEE Internet of Things Journal*, vol. 10, no. 24, pp. 21 595–21 606, 2023.

[65] J. Lee, D. Kim, and D. Niyato, “A novel joint dataset and incentive management mechanism for federated learning over mec,” *IEEE Access*, vol. 10, pp. 30 026–30 038, 2022.

[66] S. Jiang and J. Wu, “A reward response game in the federated learning system,” in *2021 IEEE 18th International Conference on Mobile Ad Hoc and Smart Systems (MASS)*, 2021, pp. 127–135.

[67] H.-S. Kang, Z.-Y. Chai, Y.-L. Li, H. Huang, and Y.-J. Zhao, “Edge computing in internet of vehicles: A federated learning method based on stackelberg dynamic game,” *Information Sciences*, vol. 689, p. 121452, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025524013665

[68] B. Han, B. Li, K. Wolter, R. Jurdak, H. Zhang, Y. Hu, and Y. Li, “Dynamic incentive design for federated learning based on consortium blockchain using a stackelberg game,” *IEEE Access*, vol. 12, pp. 160 267–160 283, 2024.

[69] W. Guo, Y. Wang, and P. Jiang, “Incentive mechanism design for federated learning with stackelberg game perspective in the industrial scenario,” *Computers Industrial Engineering*, vol. 184, p. 109592, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360835223006162

[70] C. Tang, B. Yang, X. Xie, G. Chen, M. A. Al-Qaness, and Y. Liu, “An incentive mechanism for federated learning: A continuous zero-determinant strategy approach,” *IEEE/CAA Journal of Automatica Sinica*, vol. 11, no. 1, pp. 88–102, 2024.

[71] K. Sun, J. Wu, and J. Li, “Reputation-aware incentive mechanism of federated learning: A mean field game approach,” in *2024 IEEE 9th International Conference on Smart Cloud (SmartCloud)*. Los Alamitos, CA, USA: IEEE Computer Society, may 2024, pp. 48–53. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SmartCloud62736.2024.00016

[72] S. Xuan, M. Wang, J. Zhang, W. Wang, D. Man, and W. Yang, “An incentive mechanism design for federated learning with multiple task publishers by contract theory approach,” *Information Sciences*, vol. 664, p. 120330, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025524002433

[73] J. Lu, B. Pan, A. M. Seid, B. Li, G. Hu, and S. Wan, “Truthful incentive mechanism design via internalizing externalities and lp relaxation for vertical federated learning,” *IEEE Transactions on Computational Social Systems*, vol. 10, no. 6, pp. 2909–2923, 2023.

[74] L. Cai, Y. Dai, Q. Hu, J. Zhou, Y. Zhang, and T. Jiang, “Bayesian game-driven incentive mechanism for blockchain-enabled secure federated learning in 6g wireless networks,” *IEEE Transactions on Network Science and Engineering*, vol. 11, no. 5, pp. 4951–4964, 2024.

[75] J. Han, A. F. Khan, S. Zawad, A. Anwar, N. B. Angel, Y. Zhou, F. Yan, and A. R. Butt, “Tiff: Tokenized incentive for federated learning,” in *2022 IEEE 15th International Conference on Cloud Computing (CLOUD)*, 2022, pp. 407–416.

[76] M. Ji, G. Xu, J. Ge, and M. Li, “Efficient core-selecting incentive mechanism for data sharing in federated learning,” *IEEE Transactions on Computational Social Systems*, vol. 11, no. 5, pp. 5775–5788, 2024.

[77] Y. Deng, F. Lyu, J. Ren, Y.-C. Chen, P. Yang, Y. Zhou, and Y. Zhang, “Fair: Quality-aware federated learning with precise user incentive and model aggregation,” in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, 2021, pp. 1–10.

[78] J. Huang, B. Ma, M. Wang, X. Zhou, L. Yao, S. Wang, L. Qi, and Y. Chen, “Incentive mechanism design of federated learning for recommendation systems in mec,” *IEEE Transactions on Consumer Electronics*, vol. 70, no. 1, pp. 2596–2607, 2024.

[79] Y. Yang, K. Peng, S. Wang, X. Xu, P. Xiao, and V. C. Leung, “Fairness-aware incentive mechanism for multi-server federated learning in edge-enabled wireless networks with differential privacy,” *IEEE Transactions on Mobile Computing*, pp. 1–16, 2025.

[80] X. Tang and H. Yu, “A cost-aware utility-maximizing bidding strategy for auction-based federated learning,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 36, no. 7, pp. 12 866–12 879, 2025.

[81] T. Zhou, H. Lv, N. Liu, and L. Liu, “Budget-feasible double auction mechanisms for model training services in federated learning market,” in *2024 IEEE 23rd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2024, pp. 501–507.

[82] Y. Cui, L. Yao, Y. Li, Z. Chen, B. Ding, and X. Zhou, “An auction-based marketplace for model trading in federated learning,” 2024. [Online]. Available: https://arxiv.org/abs/2402.01802

[83] E. Seo, D. Niyato, and E. Elmroth, “Auction-based federated learning using software-defined networking for resource efficiency,” in *2021 17th International Conference on Network and Service Management (CNSM)*, 2021, pp. 42–48.

[84] Y. Jiao, P. Wang, D. Niyato, B. Lin, and D. I. Kim, “Toward an automated auction framework for wireless federated learning services market,” *IEEE Transactions on Mobile Computing*, vol. 20, no. 10, pp. 3034–3048, 2021.

[85] O. Chakraborty and A. Boudguiga, “A decentralized federated learning using reputation,” *Cryptology ePrint Archive*, 2024.

[86] J. An, S. Tang, X. Sun, X. Gui, X. He, and F. Wang, “ FREB: Participant Selection in Federated Learning With Reputation Evaluation and Blockchain ,” *IEEE Transactions on Services Computing*, vol. 17, no. 06, pp. 3685–3698, Nov. 2024. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/TSC.2024.3486185

[87] J. Chen, Z. Qian, T. Meng, X. Gao, T. Wang, and W. Jia, “Fed-credit: Robust federated learning with credibility management,” 2024. [Online]. Available: https://arxiv.org/abs/2405.11758

[88] C. Ding and M. Zhang, “An overview of data contribution evaluation methods for federated learning,” *Academic Journal of Science and Technology*, vol. 11, no. 2, p. 22–26, Jun. 2024. [Online]. Available: https://drpress.org/ojs/index.php/ajst/article/view/22327

[89] M. Yao, S. Qi, Z. Tian, Q. Li, Y. Han, H. Li, and Y. Qi, “Quantifying bytes: Understanding practical value of data assets in federated learning,” *Tsinghua Science and Technology*, vol. 30, no. 1, pp. 135–147, 2025. [Online]. Available: https://www.sciopen.com/article/10.26599/TST.2024.9010034

[90] W. Li, S. Fu, F. Zhang, and Y. Pang, "Data valuation and detections in federated learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 12 027–12 036.

[91] K. D. Pandl, C.-Y. Huang, I. Beschastnikh, X. Li, S. Thiebes, and A. Sunyaev, "Scalable data point valuation in decentralized learning," *arXiv preprint arXiv:2305.01657*, 2023.

[92] Z. Wang, Z. Peng, X. Fan, Z. Wang, S. Wu, R. Yu, P. Yang, C. Zheng, and C. Wang, "Fedave: Adaptive data value evaluation framework for collaborative fairness in federated learning," *Neurocomputing*, vol. 574, p. 127227, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231223013504

[93] D. F. Groebner, P. W. Shannon, P. C. Fry, and K. D. Smith, *Business statistics*. Pearson Education UK, 2013.

[94] H. Wang, B. Zou, G. Guo, D. Yang, and J. Zhang, "Integrating trust with user preference for effective web service composition," *IEEE Transactions on Services Computing*, vol. 10, no. 4, pp. 574–588, 2015.

[95] O. A. Wahab, J. Bentahar, H. Otrok, and A. Mourad, "Optimal load distribution for the detection of vm-based ddos attacks in the cloud," *IEEE transactions on services computing*, vol. 13, no. 1, pp. 114–129, 2017.

[96] W. Saad, Z. Han, T. Basar, M. Debbah, and A. Hjorungnes, "Hedonic coalition formation for distributed task allocation among wireless agents," *IEEE Transactions on Mobile Computing*, vol. 10, no. 9, pp. 1327–1344, 2010.

[97] O. A. Wahab, J. Bentahar, H. Otrok, and A. Mourad, "Towards trustworthy multi-cloud services communities: A trust-based hedonic coalitional game," *IEEE Transactions on Services Computing*, vol. 11, no. 1, pp. 184–201, 2016.

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

[99] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE signal processing magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[100] M. K. Denko, T. Sun, and I. Woungang, "Trust management in ubiquitous computing: A bayesian approach," *Computer Communications*, vol. 34, no. 3, pp. 398–406, 2011.

[101] J. Dai, Q. Xu, W. Wang, and H. Tian, “Conditional entropy for incomplete decision systems and its application in data mining,” *International Journal of General Systems*, vol. 41, no. 7, pp. 713–728, 2012. [Online]. Available: https://doi.org/10.1080/03081079.2012.685471

[102] H. Von Stackelberg, *Market structure and equilibrium.* Springer Science & Business Media, 2010.

[103] O. A. Wahab, J. Bentahar, H. Otrok, and A. Mourad, “A stackelberg game for distributed formation of business-driven services communities,” *Expert Systems with Applications*, vol. 45, pp. 359–372, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417415006764

[104] J. Stock, O. Hauke, J. Weißmann, and H. Federrath, “The applicability of federated learning to official statistics,” in *International Conference on Intelligent Data Engineering and Automated Learning.* Springer, 2023, pp. 70–81.

[105] Y. J. Cho, D. Jhunjhunwala, T. Li, V. Smith, and G. Joshi, “To federate or not to federate: Incentivizing client participation in federated learning,” in *Workshop on Federated Learning: Recent Advances and New Challenges (in Conjunction with NeurIPS 2022)*, 2022.

[106] S. Daud, “Federated learning: Trust, fairness, and accountability,” in *Federated Learning.* CRC Press, pp. 145–159.

[107] P. M. S. Sánchez, A. H. Celdrán, N. Xie, G. Bovet, G. M. Pérez, and B. Stiller, “Federatedtrust: A solution for trustworthy federated learning,” *Future Generation Computer Systems*, vol. 152, pp. 83–98, 2024.

[108] C. Fung, C. J. M. Yoon, and I. Beschastnikh, “Mitigating sybils in federated learning poisoning,” 2020. [Online]. Available: https://arxiv.org/abs/1808.04866

[109] Z. Han, D. Niyato, W. Saad, T. Başar, and A. Hjørungnes, *Auction theory and mechanism design.* Cambridge University Press, 2011, p. 221–252.

[110] R. B. Myerson, “Optimal auction design,” *Math. Oper. Res.*, vol. 6, no. 1, p. 58–73, Feb. 1981. [Online]. Available: https://doi.org/10.1287/moor.6.1.58

[111] J. Zhang, Y. Wu, and R. Pan, “Incentive mechanism for horizontal federated learning based on reputation and reverse auction,” in *Proceedings of the Web Conference 2021*,

ser. WWW '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 947–956. [Online]. Available: https://doi.org/10.1145/3442381.3449888

[112] M. S. Ali, M. M. Ahsan, L. Tasnim, S. Afrin, K. Biswas, M. M. Hossain, M. M. Ahmed, R. Hashan, M. K. Islam, and S. Raman, "Federated learning in healthcare: Model misconducts, security, challenges, applications, and future research directions–a systematic review," *arXiv preprint arXiv:2405.13832*, 2024.

[113] Y. Tian, S. Wang, J. Xiong, R. Bi, Z. Zhou, and M. Z. A. Bhuiyan, "Robust and privacy-preserving decentralized deep federated learning training: Focusing on digital healthcare applications," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 21, no. 4, pp. 890–901, 2024.

[114] S. Moon and W. Hee Lee, "Privacy-preserving federated learning in healthcare," in *2023 International Conference on Electronics, Information, and Communication (ICEIC)*, 2023, pp. 1–4.

[115] S. Mishra, R. Tondon, and N. P. S. Rathore, "Revolutionizing healthcare with federated learning: A comprehensive review," in *2024 International Conference on Advances in Computing Research on Science Engineering and Technology (ACROSET)*, 2024, pp. 1–5.

[116] L. Zhang, J. Xu, P. Vijayakumar, P. K. Sharma, and U. Ghosh, "Homomorphic encryption-based privacy-preserving federated learning in iot-enabled healthcare system," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 5, pp. 2864–2880, 2023.

[117] X. Li, Y. Gu, N. Dvornek, L. H. Staib, P. Ventola, and J. S. Duncan, "Multi-site fmri analysis using privacy-preserving federated learning and domain adaptation: Abide results," *Medical image analysis*, vol. 65, p. 101765, 2020.

[118] V. Obarafor, M. Qi, and L. Zhang, "A review of privacy-preserving federated learning, deep learning, and machine learning iiot and iots solutions," in *2023 8th International Conference on Signal and Image Processing (ICSIP)*, 2023, pp. 1074–1078.

[119] S. V. Haldikar, O. F. M. A. Kader, and R. K. Yekollu, "Edge computing and federated learning for real-time anomaly detection in industrial internet of things (iiot)," in *2024 International Conference on Inventive Computation Technologies (ICICT)*, 2024, pp. 1699–1703.

[120] L. Yin, J. Feng, H. Xun, Z. Sun, and X. Cheng, “A privacy-preserving federated learning for multiparty data sharing in social iots,” *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 3, pp. 2706–2718, 2021.

[121] G. Long, Y. Tan, J. Jiang, and C. Zhang, *Federated Learning for Open Banking*. Cham: Springer International Publishing, 2020, pp. 240–254. [Online]. Available: https://doi.org/10.1007/978-3-030-63076-8_17

[122] D. Byrd and A. Polychroniadou, “Differentially private secure multi-party computation for federated learning in financial applications,” in *Proceedings of the First ACM International Conference on AI in Finance*, ser. ICAIF ’20. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3383455.3422562

[123] A. Imteaj and M. H. Amini, “Leveraging asynchronous federated learning to predict customers financial distress,” *Intelligent Systems with Applications*, vol. 14, p. 200064, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2667305322000059

[124] X. Ma, J. Zhu, Z. Lin, S. Chen, and Y. Qin, “A state-of-the-art survey on solving non-iid data in federated learning,” *Future Generation Computer Systems*, vol. 135, pp. 244–258, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X22001686

[125] Z. Zhang, S. Ma, J. Nie, Y. Wu, Q. Yan, X. Xu, and D. Niyato, “Semi-supervised federated learning with non-iid data: Algorithm and system design,” in *2021 IEEE 23rd Int Conf on High Performance Computing Communications; 7th Int Conf on Data Science Systems; 19th Int Conf on Smart City; 7th Int Conf on Dependability in Sensor, Cloud Big Data Systems Application (HPCC/DSS/SmartCity/DependSys)*, 2021, pp. 157–164.

[126] H. Chen and H. Vikalo, “Heterogeneity-guided client sampling: Towards fast and efficient non-iid federated learning,” in *Advances in Neural Information Processing Systems*, A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, Eds., vol. 37. Curran Associates, Inc., 2024, pp. 65 525–65 561. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2024/file/7886b9bafe76c52fd568db10ff9772df-Paper-Conference.pdf

[127] Z. Li, Z. Guan, S. Yuan, N. An, and X. Liang, "Rocfl: A robust clustered federated learning framework towards heterogeneous data," in *2023 International Conference on Intelligent Communication and Networking (ICN)*, 2023, pp. 259–264.

[128] A. Rabiee, A. Ajdarloo, and M. Rahmani, "Sgfl: A federated learning approach for non-iid data using semi-supervised dcgan," in *2023 13th International Conference on Computer and Knowledge Engineering (ICCKE)*, 2023, pp. 420–425.

[129] M. Delehouzée, X. Lessage, T. Reginster, and S. Mahmoudi, "Performance analysis of aggregation algorithms in cross-silo federated learning for non-iid data," in *2024 4th International Conference on Embedded Distributed Systems (EDiS)*, 2024, pp. 74–79.

[130] H. Wu and P. Wang, "Node selection toward faster convergence for federated learning on non-iid data," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 5, pp. 3099–3111, 2022.

[131] A. Asad, M. M. Fouda, Z. M. Fadlullah, M. I. Ibrahem, and N. Nasser, "Moreau envelopes-based personalized asynchronous federated learning: Improving practicality in network edge intelligence," in *GLOBECOM 2023 - 2023 IEEE Global Communications Conference*, 2023, pp. 2033–2038.

[132] G. Gad, Z. M. Fadlullah, M. M. Fouda, M. I. Ibrahem, and N. Nasser, "Joint knowledge distillation and local differential privacy for communication-efficient federated learning in heterogeneous systems," in *GLOBECOM 2023 - 2023 IEEE Global Communications Conference*, 2023, pp. 2051–2056.

[133] Y. Gao, Z. Chen, and N. Wan, "Personalized and similarity contrast federated learning on non-iid," in *2024 International Conference on Artificial Intelligence, Deep Learning and Neural Networks (AIDLNN)*, 2024, pp. 51–56.

[134] M. Arafeh, H. Ould-Slimane, H. Otrok, A. Mourad, C. Talhi, and E. Damiani, "Data independent warmup scheme for non-iid federated learning," *Information Sciences*, vol. 623, pp. 342–360, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025522015407

[135] C. Herath, X. Liu, S. Lambotharan, and Y. Rahulamathavan, "Enhancing federated learning convergence with dynamic data queue and data entropy-driven participant selection," *IEEE Internet of Things Journal*, pp. 1–1, 2024.

[136] R. D. Gupta and D. S. P. Richards, "The history of the dirichlet and liouville distributions," *International Statistical Review / Revue Internationale*

de Statistique, vol. 69, no. 3, pp. 433–446, 2001. [Online]. Available: http://www.jstor.org/stable/1403455

[137] S. Caldas, P. Wu, T. Li, J. Konečný, H. B. McMahan, V. Smith, and A. Talwalkar, "Leaf: A benchmark for federated settings," *arXiv preprint arXiv:1812.01097*, 2018. [Online]. Available: https://arxiv.org/abs/1812.01097

[138] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, J. Fernandez-Marques, Y. Gao, L. Sani, H. L. Kwing, T. Parcollet, P. P. d. Gusmão, and N. D. Lane, "Flower: A friendly federated learning research framework," *arXiv preprint arXiv:2007.14390*, 2020.

[139] X. Ouyang, Z. Xie, J. Zhou, J. Huang, and G. Xing, "Clusterfl: a similarity-aware federated learning system for human activity recognition," in *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 54–66. [Online]. Available: https://doi.org/10.1145/3458864.3467681

[140] X. Li, M. Jiang, X. Zhang, M. Kamp, and Q. Dou, "Fedbn: Federated learning on non-iid features via local batch normalization," 2021. [Online]. Available: https://arxiv.org/abs/2102.07623

[141] R. Lee, M. Kim, D. Li, X. Qiu, T. Hospedales, F. Huszár, and N. D. Lane, "Fedl2p: Federated learning to personalize," 2023. [Online]. Available: https://arxiv.org/abs/2310.02420