TD:04/2025

FEDERAL UNIVERSITY OF PARÁ
INSTITUTE OF TECHNOLOGY
POSTGRADUATE PROGRAM IN ELECTRICAL ENGINEER

ALEX BARROS DOS SANTOS

LOCAL UPDATES AND CLIENT SELECTION STRATEGIES FOR FEDERATED LEARNING SYSTEMS

UFPA / ITEC / PPGEE
Guamá University Campus
Belém-Pará-Brazil

2025

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

"LOCAL UPDATES AND CLIENT SELECTION STRATEGIES FOR FEDERATED LEARNING SYSTEMS"

AUTOR: **ALEX BARROS DOS SANTOS**

TESE DE DOUTORADO SUBMETIDA À BANCA EXAMINADORA APROVADA PELO COLEGIADO DO PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA, SENDO JULGADA ADEQUADA PARA A OBTENÇÃO DO GRAU DE DOUTOR EM ENGENHARIA ELÉTRICA NA ÁREA DE COMPUTAÇÃO APLICADA.

APROVADA EM: 10/03/2025

BANCA EXAMINADORA:

Prof. Dr. Eduardo Coelho Cerqueira
(Orientador – PPGEE/ITEC/UFPA)

Prof. Dr. Denis Lima do Rosário
(Avaliador Interno - PPGEE/ITEC/UFPA)

Prof. Dr. Iago Lins de Medeiros
(Avaliador Externo ao Programa - CAMPUS TUCURUÍ/UFPA)

Prof. Dr. Allan Douglas Bento da Costa
(Avaliador Externo - UFRA)

Prof. Dr. Hugo Leonardo Melo dos Santos
(Avaliador Externo - UEPA)

VISTO:

Prof. Dr. Diego Lisboa Cardoso

(Coordenador do PPGEE/ITEC/UFPA)

Dados Internacionais de Catalogação na Publicação (CIP) de acordo com ISBD
Sistema de Bibliotecas da Universidade Federal do Pará
Gerada automaticamente pelo módulo Ficat, mediante os dados fornecidos pelo(a) autor(a)

B277l Barros dos Santos, Alex.
Local updates and client selection strategies for federated learning systems / Alex Barros dos Santos. — 2025.
84 f. : il. color.

Orientador(a): Prof. Dr. Eduardo Coelho Cerqueira
Coorientador(a): Prof. Dr. Prof. Dr. Denis Lima do Rosário
Tese (Doutorado) - Universidade Federal do Pará, Instituto de Tecnologia, Programa de Pós-Graduação em Engenharia Elétrica, Belém, 2025.

1. Federated Learning. 2. Non-iid. 3. Client Selection. 4. IoT. 5. Communication efficiency. I. Título.

CDD 621.3

TD:04/2025

FEDERAL UNIVERSITY OF PARÁ
INSTITUTE OF TECHNOLOGY
POSTGRADUATE PROGRAM IN ELECTRICAL ENGINEER

ALEX BARROS DOS SANTOS

LOCAL UPDATES AND CLIENT SELECTION STRATEGIES FOR FEDERATED LEARNING SYSTEMS

Thesis proposal submitted to the PhD qualifying examination in Electrical Engineering at the Federal University of Pará.

Advisor: Dr. Eduardo Coelho Cerqueira

Co-advisor: Dr. Denis Lima do Rosário

UFPA / ITEC / PPGEE
Guamá University Campus
Belém-Pará-Brazil

2025

TD:04/2025

ALEX BARROS DOS SANTOS

LOCAL UPDATES AND CLIENT SELECTION STRATEGIES FOR FEDERATED LEARNING SYSTEMS

Thesis proposal submitted to the PhD qualifying examination in Electrical Engineering at the Federal University of Pará.

Approved in: 10/March/2025

JUDGING PANEL

Ph.D. Eduardo Coelho Cerqueira
(Federal University of Pará - Advisor)

Prof. Dr. Denis Lima do Rosario
(Federal University of Pará - Co-Advisor)

Prof. Dr. Allan Douglas Bento da Costa
(Federal Rural University of Amazonia - External Member)

Prof. Dr. Iago Lins de Medeiros
(Federal University of Pará - Internal Member)

Prof. Dr. Hugo Leonardo Melo dos Santos
(State University of Pará - External Member)

Abstract

Local updates and client selection strategies for federated learning systems

Advisor: Eduardo Coelho Cerqueira
Key words: Federated Learning; Non-iid; Client Selection; IoT; Communication efficiency.

Federated Learning (FL) has emerged as a promising approach for enabling collaborative machine learning across distributed clients while preserving data privacy. However, FL systems face significant challenges due to non-IID data distributions, system heterogeneity, and communication efficiency limitations. This thesis proposes novel strategies to enhance the performance of FL through adaptive local update schemes and intelligent client selection mechanisms. First, we conduct an empirical evaluation of the impact of local update configurations on the convergence and generalization of popular FL algorithms, namely FedAVG and FedADAM. Our analysis reveals that increasing the number of local training epochs can accelerate convergence but also risks model overfitting, thereby reducing generalization. To mitigate this trade-off, we propose an adaptive controller that dynamically adjusts the number of local epochs using a Poisson distribution. Second, we introduce MESFLA (Model Efficiency through Selective Federated Learning Algorithm), a cluster-based client selection mechanism that considers both model weights and data size characteristics to strategically select participants for each FL training round. MESFLA employs the CKA (Centered Kernel Alignment) algorithm to cluster clients based on model weight similarity and then selects the most relevant clients from each cluster. The adaptive local update controller dynamically balances convergence speed and model generalization, while MESFLA's intelligent client selection optimizes performance in heterogeneous data settings. MESFLA consistently achieves higher model accuracy, faster convergence, and requires fewer communication rounds between clients and the server, demonstrating its effectiveness in non-IID data environments typical of real-world FL scenarios.

Contents

List of Abbreviations

6G	Sixth Generation Wireless Networks
AI	Artificial Intelligence
AR	Augmented Reality
CKA	Centered Kernel Alignment
CV	Computer Vision
DM	Data Mining
DNNs	deep neural networks
FedAVG	Federated averaging
FL	Federated Learning
FLSPs	FL System Parameters
GDPR	General Data Protection Regulation
HFL	Horizontal Federated Learning
IID	Independent and Identically Distributed
IoT	Internet of Things
LGPD	General Law of Data Protection
MEC	Multi-access edge computing
MESFLA	Model Efficiency through Selective Federated Learning Algorithm
ML	Machine Learning

MTL	Multi-Task Learning
NLP	Recommendation Systems
non-IID	non-Independent and non-Identically Distributed
RS	deep neural networks
SGD	Stochastic Gradient Descent
VFL	Vertical Federated Learning
VR	Virtual Reality

List of Figures

List of Tables

CHAPTER 1

Introduction

This thesis proposal presents strategies for improving Federated Learning (FL) system performance through local weight updates and client selection techniques. FL algorithms face significant challenges arising from factors such as Independent and Identically Distributed (IID) and non-IID data distributions, device heterogeneity in communication and computational capabilities, machine learning model configurations, and varying client numbers. This chapter presents key concepts of FL applications and their challenges related to FL System Parameters (FLSPs) and their performances. Furthermore, it motivates this research work and establishes the research questions, objectives, contributions, and thesis structure.

1.1 Overview

Emerging technologies, such as augmented reality (AR), virtual reality (VR), the Internet of Things (IoT), and social networking applications, have led to unprecedented growth in the volumes of data generated daily. Access to data enables better products, smarter models, and creates new products such as the metaverse, where VR and AR are fundamental enabling technologies. Recent developments include chatbot applications powered by Large Language Models[6]. A whole new world of data sharing and production is on the horizon. As more data is generated, the need for transmission has driven significant advancements in wireless mobile communication systems. One of the most significant changes has been the shift from centralized to decentralized approaches. Initially, early devices had significant limitations: low battery life, insufficient storage capacity, and limited computational power. These constraints led to the adoption of centralized cloud centers within 5G communication systems, designed to aggregate, store, and analyze large volumes of data.

However, the massive proliferation of smart devices and the exponential growth in data traffic have introduced substantial challenges to centralized and decentralized systems in 5G and earlier generations of wireless networks communication systems. Key among these challenges is the high latency in processing, storing, and transmitting vast amounts of mobile data, particularly in ultra-dense networks where bandwidth constraints are a critical bottleneck.

To address these issues, Multi-access edge computing (MEC) has emerged as a decentralized solution. MEC is an ETSI-standardized architecture that brings computing resources, storage capabilities, and application services to the network edge, in close proximity to IoT devices. By enabling local computational operations and data storage at the edge, MEC reduces dependence on distant central servers, minimizes communication delays, and supports real-time applications with ultra-low latency requirements while conserving network bandwidth.

Besides communication issues, organizations must rethink their data governance, retention, and data privacy policies. For example, the International Data Corporation (IDC)[7] predicted that by 2025, 79 ZB of data will be created by billions of IoT devices. Data privacy has drawn attention from several governments, raising serious concerns about data centralization in cloud servers. Various legislation, such as the U.S. Consumer Privacy Bill of Rights, the European Commission's General Data Protection Regulation (GDPR), and the General Law of Data Protection (LGPD) in Brazil, have been designed to protect users' privacy, particularly within medical domains[1]. High propagation delays to cloud servers and insufficient capacity to meet delay-sensitive application requirements have motivated the deployment of edge computing. This approach aims to bring cloud services to the network edge close to the end-user[8].

FL is defined as a machine learning setting where multiple entities (clients or devices) collaborate in solving a machine learning problem, under the coordination of a central server or service provider [9]. As many regulations prohibit raw data transfer, FL emerged as a promising technique where sensitive data remains preserved locally and only the model's parameters are transferred through the network. FL brings together many research communities such as cryptography, databases, and machine learning. These communities study how to analyze and learn from data distributed among many owners without exposing that data.

This idea of training models without collecting raw training data in a single location has proven to be useful in other practical scenarios, for example, learning from data silos such as businesses or medical institutions which cannot share data due to confidentiality or legal constraints, or applications in edge networks [10]. Wang et al[11] discussed how FL has been adopted by many companies besides its founder, Google, which makes extensive use of FL in the Gboard mobile keyboard for applications including next word prediction, emoji suggestion and out-of-vocabulary word discovery.

Apple has implemented FL in iOS 13 for applications like the QuickType keyboard and the vocal classifier for "Hey Siri". WeBank has applied it for money laundering

detection, while Intel and Consilient use it for financial fraud detection. In the medical field, NVIDIA has employed FL for predicting COVID-19 patients' oxygen needs[12],, thereby avoiding sensitive data transmission. It also enables the federation of sensors, appliances, and smart devices in smart city scenarios. These devices can collaborate to train models for city governance. Smart vehicles can exploit FL-trained models for autonomous driving, predicting traffic flows, and planning routes.

1.2 Motivations and Challenges

FL was proposed in 2016 and rapidly gained attention from both academia and industry. Liu et al.[13] conducted a survey of articles published from 2019 to 2022 in top-tier conferences, and during the course of this thesis, I updated the count to include those presented at the INFOCOM conference. This growth in interest correlates with the success of deep neural networks (DNNs), which have received significant attention due to their remarkable performance on various tasks such as Computer Vision (CV), Natural Language Processing (NLP), Recommendation Systems (RS), and Data Mining (DM). However, access to data remains challenging due to storage limitations and collection difficulties. Furthermore, data held by individual parties may be sensitive or contain user privacy information. An important and practical challenge in real-world scenarios is how to aggregate insights from different parties while ensuring privacy protection.

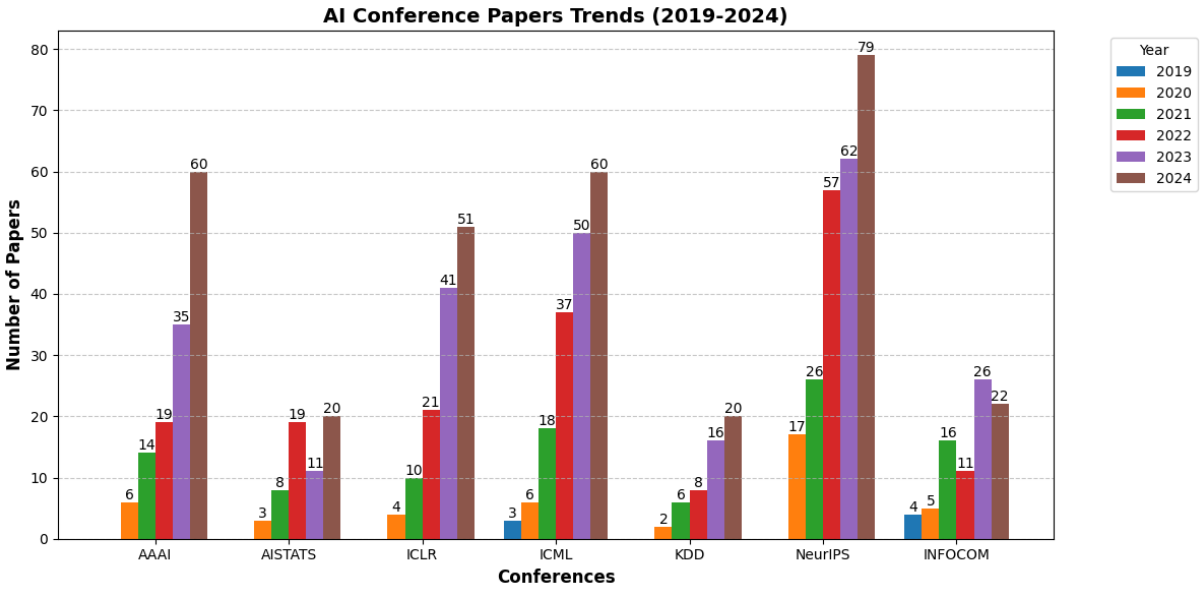


Figure 1: The number of published FL papers in top-tier conference from 2019-2024

A central motivator for FL is to provide strong privacy protection. Storing data locally, rather than replicating it in the cloud, decreases the system's attack surface; however, integration with other technologies such as differential privacy and secure multiparty computation (SMPC) protocols is essential. These privacy technologies put additional constraints on the architectural framework for federated optimization algorithms. The

sixth generation (6G) of wireless networks represents the next evolution in telecommunications technology, promising terabit-per-second data rates, sub-millisecond latency, and seamless three-dimensional coverage including air, sea, and space. The integration of Artificial Intelligence (AI) and Machine Learning (ML) into 6G networks offers enhanced network management orchestration performance by autonomously addressing optimization challenges. As an example, AI can enhance network power efficiency by dynamically activating and deactivating components in response to real-time operational conditions, thereby eliminating human-induced errors. FL enables model training on users' local devices, subsequently transmitting only model parameters to the server to protect sensitive raw data from potential exposure to untrustworthy servers[1].

FL has extensive applications in various aspects of our lives, such as:

- Aerial and non-terrestrial networks: FL is crucial for non-terrestrial networks, including aerial systems, supporting the development of robust communication systems to improve connectivity, resource management, and real-time data processing in remote and hard-to-reach areas. FL offers significant potential for enhancing the performance and efficiency of aerial networks, including Unmanned Aerial Vehicles (UAVs), High Altitude Platforms (HAPs), Low Altitude Platforms (LAPs), and Low Earth Orbit (LEO) satellites. Through decentralized data processing and model training, FL enables these systems to collaboratively learn from vast amounts of data while maintaining privacy and security. The combination of FL and UAVs advances fields such as IoT applications, intelligent transportation systems, and environmental monitoring, while ensuring data security and efficiency in dynamic environments[1].

- Healthcare Systems: The rapid expansion of smart healthcare systems has made early detection of a wide variety of chronic diseases more accessible and cost-effective. Medical facilities are shifting toward remote diagnosis and treatment, leading to increased use of IoT-enabled medical equipment. FL enables the establishment of secure healthcare systems, ensuring the privacy and security of medical data against various threats. The proliferation of medical sensors and wearable technology necessitates solutions that enable collaborative development of health-related predictive algorithms while preserving individual health data privacy.

- Smart Cities and homes: FL optimizes urban services such as traffic management and energy efficiency while preserving data privacy in smart city scenarios. In smart homes, FL enables personalized automation without compromising user privacy while improving comfort and energy efficiency. The deployment of numerous sensors and data-gathering devices creates multiple applications for FL where energy efficiency and data protection are priorities, including traffic management, public safety, environmental monitoring, urban planning, and air quality assessment.

FL is an interdisciplinary research field, and considerations such as fairness and personalization are also of interest. FL must contend with many key issues that centralized

methods typically do not address. These challenges include:

- Communication efficiency: As with any kind of distributed computing, reducing communication between server and clients is crucial for system efficiency. This can become the main bottleneck when dealing with numerous clients/devices. For mobile applications using IoT devices, bandwidth is usually limited and connection availability must be considered. To address these challenges, researchers have proposed compression mechanisms for reducing communication costs and improved optimization methods to reduce the total number of communication rounds. One approach, also used in this thesis, involves taking multiple local updates on devices' data before global synchronization, which often achieves good results in practice.

- Heterogeneity: This challenge manifests in multiple aspects: data, system, and model heterogeneity. Data generated and stored at the edge is typically imbalanced (training data quantity varies significantly between clients) and statistically heterogeneous. The non-Independent and non-Identically Distributed (non-IID) data partitioning in federated settings presents challenges when training a single global model for all clients, particularly under limited communication budgets. For example, identical object images collected from different environments, or the same activity from different people, can lead to varying data distributions, affecting federated aggregation performance [14]. Model heterogeneity arises when clients use different model architectures to optimize specific tasks. System heterogeneity occurs due to clients having different hardware, manufacturers, storage space, computational power, and communication capabilities. The server must often decide whether to wait for all client model updates for better accuracy or proceed with partial participation.

- Privacy and security: FL's fundamental principle requires algorithms to maintain raw data locally and preserve privacy. FL can be combined with other privacy technologies, such as differential privacy [15] and secure aggregation, to enhance system security. For many applications, federated optimization methods must be compatible with such privacy protection methods or achieve comparable privacy guarantees[11]. Measures must be implemented to prevent malicious attackers from inferring properties of raw data through various schemes.

- System complexity: Practical FL systems often involve complex architectures, such as hierarchical aggregation between the coordinating server in the data center and client edge devices. Various protocols designed to build such systems[16] can introduce implicit constraints for federated optimization. For instance, stragglers (devices slow to report updates) influence decisions between synchronous and asynchronous optimization approaches. The robustness of the system depends on multiple factors, including heterogeneity of the device and the data.

Kairouz et al.[9] defined FL in two different settings namely, cross-silo and cross-device. Cross-silo FL usually occurs by the exchange of machine learning models between

institutions assuming reliable network connection and a more controlled environment. Cross-device FL involves thousands of edge-devices participating in the federation. In this setting, usually, a small fraction of clients participate in each round of model training, clients cannot maintain state across rounds most of the time, and there are constraints related to network communication, mobility, and edge computing [17].

There are many practical applications of the general idea of FL. We highlight two settings: cross-device FL and cross-silo FL. Cross-device FL targets machine learning across large populations of mobile devices, while cross-silo FL targets collaborative learning among several organizations. Both cross-device and cross-silo federated optimization have data and computational heterogeneity as well as communication and privacy constraints, but they also have some differences that we should highlight[11]:

- Number of clients: Cross-silo FL and cross-device FL differ significantly in client population. In cross-silo FL, each organization represents one client, typically ranging from two to several dozen clients. Conversely, cross-device FL can involve millions of clients, corresponding to individual devices for a single application. However, the volume of private data per client in cross-device FL tends to be smaller than in cross-silo FL
- Availability: Cross-silo clients, typically hosted in data center organizations, maintain high availability, enabling consistent participation in each round. Cross-device settings, however, involve a large population of intermittently-available devices. The server must implement a sampling process to access clients, with limited control over client selection and participation.
- Computation and communication constraints: Cross-device settings face stricter computation and communication constraints. Edge devices typically have limited computational power and communication capabilities, whereas cross-silo clients, often based in data centers, can utilize hardware accelerators and high-bandwidth communication links.

In conclusion, FL is a distributed approach the can be used in many applications scenarios, allowing us to take advantage of data generated from several clients and preserving the privacy required today. However, as we mentioned, FL is not a silver bullet to unlock the indiscriminate use of data in the edge. It needs several aspects to be addressed, both for small and larger scale applications to established it as a lasting solution for next years.

1.3 Research Questions

Based on the above motivation, we considered the following research questions for this thesis:

- How to select efficient FL algorithms for scenarios with a large number of clients?
- How do data distribution and dataset types impact FL system performance?
- What are the most relevant parameters and metrics for evaluating FL applications?
- How to improve client selection mechanisms in non-IID data scenarios?
- How to achieve model efficiency and fast convergence by considering model and data size information in client selection mechanisms?

1.4 Objectives

In this thesis proposal, we investigate cross-device scenarios where communication-efficient protocols still need to be developed. In a cross-device setting, each client maintains a local training dataset that is never uploaded to the server[15]. Instead, clients compute local updates of model parameters that are communicated to the server, reducing privacy and security risks by limiting the attack surface to individual devices rather than all devices and the cloud. Standard optimization methods commonly used in machine learning, such as distributed Stochastic Gradient Descent (SGD), may incur large communication overhead when frequently communicated to the central server. To address this, clients perform multiple local updates before communicating model parameters to the server, significantly reducing the required communication for model training[17]. Federated Averaging (FedAVG) is a baseline algorithm that aggregates parameters from multiple clients at the central server. Although widely used, Karimireddy et al.[18] demonstrated the convergence issues that FedAVG faces.

We explored FedOPT, proposed by Reddi et al.[17],a flexible algorithmic framework that enables clients and servers to employ optimization methods beyond the SGD method used in FedAVG. However, few studies have explored this framework or empirically evaluated its effectiveness regarding communication efficiency issues[19] [20]. Additionally, we investigated client selection mechanism challenges in non-IID data scenarios. Our research assessed clustering algorithms to select clients with higher relevance in each group—those that contribute significantly to model training in each learning round—thereby improving accuracy and convergence. We propose a cluster-based client selection mechanism for FL applications that considers model and data size informationy, called MESFLA.

The objectives of this research are:

- Develop and implement communication-efficient algorithms (such as FedAVG and FedOPT) for FL applications.
- Establish optimized parameter configurations for different data distributions and types in FL systems.

- Design performance metrics and adaptive techniques for FL algorithm selection.
- Develop a clustering-based client selection mechanism that improves handling of non-IID data.
- Implement a monitoring system at the aggregation server that evaluates model convergence and efficiency considering data and architecture size.

1.5 Contributions

The main contributions are:

1. A comprehensive evaluation of local adaptation strategies in FL, demonstrating how multiple local updates enhance model accuracy while reducing communication rounds.
2. Development of a dynamic adaptation mechanism that optimizes the trade-off between accuracy, communication efficiency, and different data distributions.
3. Identification and application of relevant parameters and metrics for FL system evaluation.
4. Design and implementation of a clustering-based client selection mechanism (MES-FLA) for non-IID data scenarios.
5. A detailed performance evaluation demonstrating how our approach maintains rapid convergence while considering model and data size characteristics.

1.6 Text Organization

This text presents the fundamentals of this research based on related works, the work of one paper published in a conference focused in local updates to avoid over fitting, a second paper published in a journal about client selection strategies, and the ongoing research being developed by the group research as future works. The remaining of this document is structured as follow:

- Chapter 2: Presents the fundamental concepts of FL and prominent algorithms. It describes the main datasets used in simulations and discusses key challenges in client selection strategies that affect the overall performance of FL systems.
- Chapter 3: Reviews the current state-of-the-art literature relevant to the thesis proposal.

- Chapter 4: Presents our comprehensive evaluation of local adaptation strategies in FL, demonstrating how multiple local updates enhance model accuracy while reducing communication rounds. This chapter details our experiments with FedAVG and FedOPT algorithms using the Adam optimizer, and introduces our dynamic adaptation mechanism for variable epoch numbers.

- Chapter 5: Details our proposed cluster-based client selection mechanism for FL applications, incorporating model architecture and data size characteristics. This chapter presents performance evaluations demonstrating rapid convergence and reduced communication rounds between clients and the aggregation server.

- Chapter 6: Concludes the thesis proposal, outlines expected future work, and presents the published works associated with this project.

CHAPTER 2

Fundamental Concepts

This chapter introduces fundamental concepts of federated learning, initially examining the architectural definitions of centralized and distributed learning that led to federated learning's development. We then detail the foundational FedAvg algorithm, its variations, and advanced approaches including FedProx, FedMA, FedOpt, FedAdam, and FedMoon. Given the challenges in acquiring real-world datasets for federated learning research, we present key benchmark datasets commonly used in FL simulations, such as MNIST, CIFAR-10, and Stack Overflow. The chapter concludes with an analysis of client selection techniques that enhance federated learning performance across diverse mobile and heterogeneous scenarios.

2.1 Centralized learning

Centralized learning represents the most straightforward approach to training machine learning models, relying on a single server or cluster to perform all computations. This architecture gathers data from various sources and facilitates simplified resource allocation and coordination. A key advantage is centralized access to the entire dataset, ensuring consistency and standardization in model training, which can lead to improved model accuracy and performance.

This approach proves particularly advantageous in scenarios where computational resources are abundant and data privacy concerns are minimal or can be managed effectively. For example, industries with dedicated high-performance computing infrastructure or those handling non-sensitive data can leverage centralized learning to achieve faster deployment and robust model performance. However, centralized learning faces several significant limitations. The primary drawback is its reliance on a single point of computation, which reduces scalability when handling large datasets or extensive client networks.

The centralized architecture poses substantial risks to data privacy, requiring aggregation of all data in one location and potentially exposing sensitive information to security breaches. Furthermore, dependence on a central server creates a processing bottleneck, potentially leading to increased latency or system failure during server outages.

These scalability and privacy challenges, while manageable in certain contexts, demonstrate the necessity for exploring decentralized alternatives in modern machine learning systems.

2.2 Decentralized learning

Decentralized learning distributes the training process across multiple nodes or devices, eliminating the need for a central server to manage or aggregate data. This architecture inherently prioritizes privacy by keeping data localized to the nodes where it is generated, reducing the need for extensive data transfers. This approach also supports scalability, making it suitable for environments with numerous devices or clients.

A significant advantage of decentralized learning is its robustness. By distributing the workload among multiple nodes, the system becomes less vulnerable to single points of failure, ensuring training continuity even when nodes fail or experience network issues. This resilience makes decentralized learning particularly appealing in critical domains such as healthcare and finance, where data sensitivity and uninterrupted operation are essential.

Despite its strengths, decentralized learning presents notable challenges. Coordinating the learning process across diverse and geographically distributed nodes can be complex, particularly when nodes have heterogeneous resources or data distributions. Maintaining process security and ensuring effective model convergence across nodes require sophisticated synchronization and optimization strategies. Furthermore, the absence of a central aggregation mechanism can lead to training inefficiencies and difficulties in achieving global model consistency.

FL emerges as a compelling framework that bridges the gap between centralized and fully decentralized approaches, introducing structured mechanisms for coordinating learning across nodes while preserving privacy and addressing many challenges associated with fully decentralized systems. The following section examines FL's principles, advantages, and its evolution from decentralized learning foundations to create scalable and secure solutions for modern machine learning applications.

2.3 Federated Learning

FL integrates the strengths of decentralized and centralized approaches, enabling collaborative model training across distributed devices while maintaining data privacy. In

this paradigm, data remains decentralized, generated, and stored locally on the devices or clients where it originates. Each device retains a unique set of local data, avoiding the need for raw data to be shared or centralized. This federated approach ensures privacy-preserving updates by transmitting only model parameters, such as weights and gradients, rather than raw data. These updates are periodically aggregated on a central server to refine a global model, creating a unified learning outcome while preserving individual privacy. The federated architecture enables adaptive personalization by tailoring the global model to specific user contexts without compromising sensitive data. The federated learning framework is particularly suited to applications where data privacy is critical, such as personalized recommendation systems, predictive maintenance, healthcare, and financial services. By keeping raw data on the device or with the user, this approach enhances security and minimizes exposure to potential breaches. Furthermore, its sophisticated aggregation methods protect individual contributions within the global learning process while maintaining model performance and accuracy.

Through this innovative framework, FL empowers devices to contribute to collective intelligence without sacrificing data sovereignty, offering a scalable and secure solution for modern machine learning challenges [8]. 2 demonstrates the three approaches discussed in this chapter.

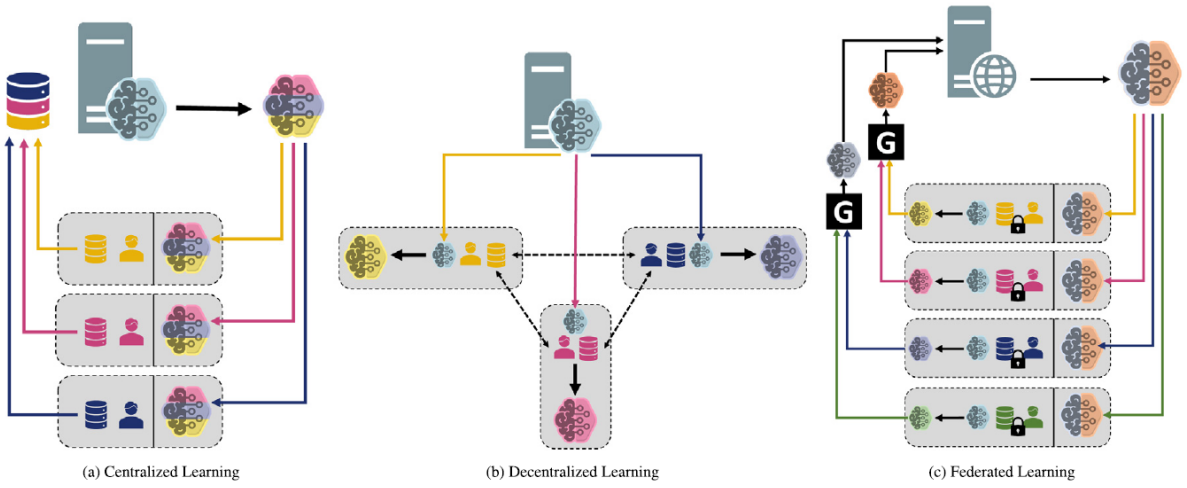


Figure 2: Comparison of three centralized learning, decentralized learning, and federated learning structures [1]

Federated Learning (FL) emerged as a paradigm that bridges the transition from centralized (cloud-based) to distributed on-device learning (edge server), where collected data remains on local devices and servers for training local models, safeguarding data privacy and confidential information[1]. This decentralized approach offers multiple benefits: offline operability, enhanced latency performance and accuracy, privacy fortification, prolonged battery life, and localized model training. In this framework, knowledge accumulation occurs through model parameter sharing rather than raw data transfer, significantly reducing communication overhead. The accumulated knowledge is subsequently aggregated and disseminated among participants through federated model sharing on a

central server. The following section describes the FL algorithm in detail.

2.3.1 Basic Algorithm

The general idea of FL is depicted in Figure 3. It consists of one central server called parameter server and a set of N clients, each having its local dataset. At the beginning of each federated training iteration, a subset C $\subseteq$ N of clients is chosen to receive the current global state of the shared model in terms of model weights. This model is broadcasted to clients (step 1). Each client uses its CPU and energy resources to carry out local computations on its dataset based on the shared parameters (step 2). Clients then send the model updates (step 3) to the parameter server which applies these updates to its current global model to generate a new one (step 4).

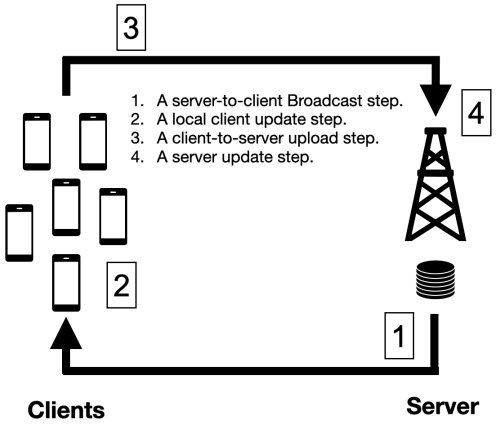


Figure 3: The overview of four main steps of FL Training Process

This process is repeated over several iterations (sometimes referred to as epochs) until the global model reaches a certain accuracy level determined by the parameter server. In summary, an FL scenario consists of two main phases, i.e., local update and global aggregation. The local update phase refers to the process of computing the gradient descents by the client devices to minimize the underlying loss function for their local data. Global aggregation includes the steps of collecting the updated model parameters by the server from the different client devices, aggregating these parameters, and then sending back the aggregate parameters to the clients to be used in their next training iteration.

In terms of optimization research, there are opportunities in client and server optimization. The former is related to the process of local updates/training while the later is related to the process aggregating the model parameters sent by clients to produce an enhanced model. FL needs some orchestration back and forth between the server and the devices to train and evaluate a model, there are many architecture and strategies related to this orchestration. The initial model can be either initialized randomly or can be pre-trained. This process happens hundreds or even thousands of times in model training. The service provider is interested in the data that the device has. This process is going to happen hundreds or maybe thousands of times. Both the initial model and how the

model evolves are important and must get attention from the management entity that controls all this process. To determine the optimal set of parameters that fits the training data, the training model has to optimize a loss (objective) function, which penalizes the model when it produces an inaccurate label on a data point.

Stochastic gradient descent is the backbone of ML and for large datasets, it is implemented distributed where the dataset is shuffled and split equally across worker nodes. Ideally, if we have more nodes, we are processing more data per iteration and we expect to increase the speed to get a target accuracy, but in reality, it is not so easy to achieve such speed due to issues related to synchronization and communication delays that increase with the number of clients. The problems related to synchronization delays are related to workers with less computing power as demonstrated by Dutta et al.[21]. Communication delay is the time taken to aggregate the gradients, update them, and send the updated model back to the workers/clients.

One solution to overcome this communication delay is called local update SGD, basically, the workers perform more local updates instead of computing only one update and send the model to the server aggregates. This reduces the frequency of communication and makes the derivation of a single model more efficient. However, clients are not homogeneous as expected and they are not available for training all the time. Then, just add more clients does not bring the process to convergence. If too many clients drop off the training process can become unstable. In the Following subsections we describe some algorithms used in FL problems.

2.4 Categories of FL

FL can be categorized into horizontal FL (HFL) and vertical FL (VFL) based on data distribution characteristics. In HFL, datasets share the same feature space but differ in sample space, making it suitable for institutions with similar data structures, such as hospitals or banks. In VFL, datasets overlap in sample space but differ in features, enabling collaboration between organizations with complementary data, such as healthcare providers and insurers.

2.4.1 Vertical Federated Learning

Vertical Federated Learning (VFL) addresses scenarios where data is partitioned vertically—features are distributed across multiple parties while sample IDs partially overlap. This approach enables organizations to collaboratively train machine learning models without sharing raw data, making it particularly effective for scenarios with heterogeneous features across entities. To illustrate VFL's application, consider diabetes prediction in healthcare. Research shows that high blood pressure and obesity increase Type-2 diabetes risk[2]. While healthcare providers may have access to patient weight, age, and medical history, they might lack data about lifestyle factors such as diet and physical activity.

VFL enables secure collaboration between healthcare providers and fitness tracking applications, combining their complementary datasets without exposing sensitive information, as demonstrated in Fig. 1.

Patient data stored across healthcare organizations provides another compelling example of VFL application. Privacy regulations typically prevent direct merging of patient datasets from different organizations for model training. VFL circumvents this limitation by enabling collaborative training on distributed datasets while maintaining data within secure premises. This approach presents unique challenges in entity resolution and feature alignment, making traditional centralized aggregation methods unsuitable. Yurdem el al.[2] demonstrates VFL's significant potential for advancing machine learning in privacy-sensitive domains. Current implementations show promising results, particularly in healthcare and financial sectors, while ongoing research focuses on optimizing VFL for complex machine learning architectures and improving its performance across diverse application domains. Fig 4 describes an architecture of VFL application.

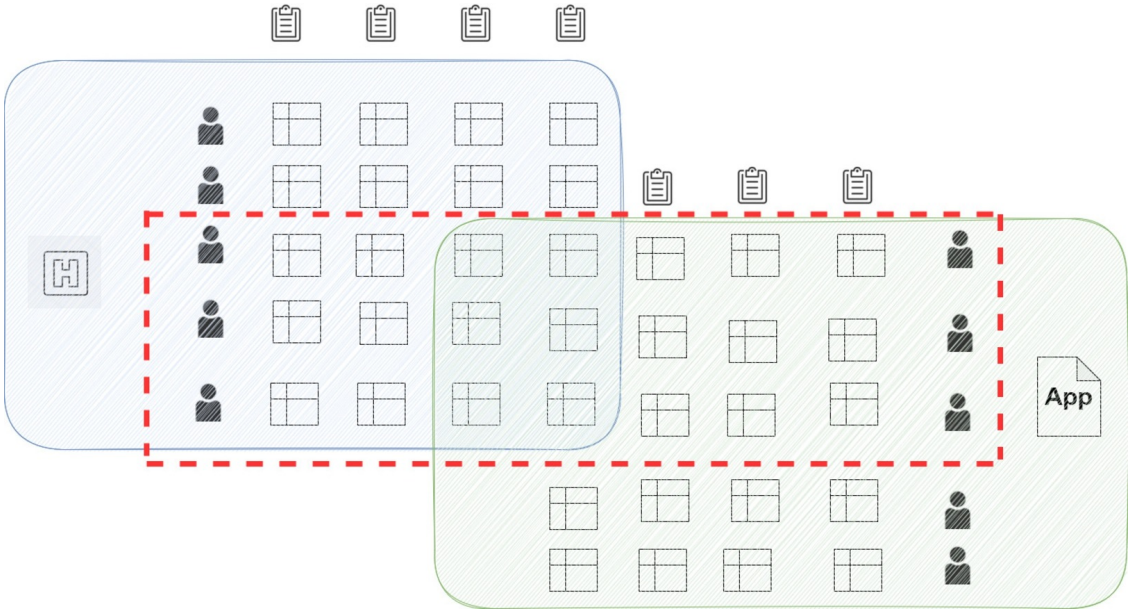


Figure 4: An example of vertical FL-based application [2].

2.4.2 Horizontal Federated Learning

In horizontal Federated Learning (HFL), data features across different nodes remain similar while sample spaces differ significantly. This approach is predominantly implemented in IoT and smart device networks, where data shares common features but is distributed across diverse user devices or geographical locations.

To address specific challenges, such as the scarcity of labeled data, a hierarchical heterogeneous HFL framework has been proposed. This approach leverages heterogeneous domains to mitigate label shortages by adapting each participant's dataset as a target domain. By doing so, it facilitates efficient learning even in scenarios with limited labeled

entities.

In practical applications, such as healthcare, the complexity of data collection poses significant challenges. It is often infeasible for individual hospitals or organizations to establish centralized data pools due to privacy concerns or logistical limitations. Horizontal FL provides an effective solution by enabling multiple parties to collaboratively train a machine learning model using similar datasets sourced from different institutions, without sharing raw data. This ensures both data privacy and robust model training.

A real-world application of horizontal FL in healthcare involves hospitals collaboratively training a predictive model for disease diagnosis while keeping patient data decentralized. This approach not only safeguards privacy but also improves model generalization by leveraging diverse datasets. An example of horizontal FL in action is illustrated in Figure 5, showcasing its utility and adaptability across distributed learning environments.

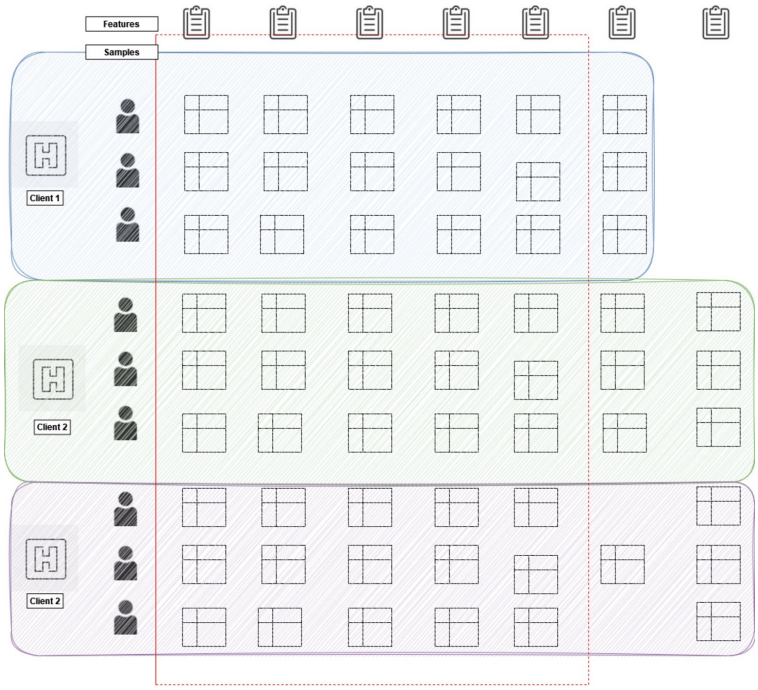


Figure 5: An example of horizontal FL-based application [2].

2.4.3 FedAVG and Weighted federated averaging

Federated Averaging (FedAvg) is a global model aggregation algorithm executed by the parameter server upon receiving local updates from clients. As shown in Algorithm 1 (Figure 7), the process begins with server initialization of the training model parameters (line 3), followed by selection of eligible clients for participation (line 5). Each selected client then performs local training by minimizing their local loss function using Stochastic Gradient Descent (SGD) on algorithm-selected mini-batches from the training dataset (line 7). Following receipt of model updates from participating clients, the server

minimizes the global loss function by averaging the received gradients. This federated training process continues iteratively until the global loss function converges or achieves the desired accuracy threshold. The research conducted by Li et al.[22] reveals an important observation: multiple epochs (E) of local updates with a small learning rate produce effects similar to a single epoch with a larger learning rate. Additionally, the authors demonstrate that partial client participation—where not all selected clients complete the training process—remains manageable, though it increases variance in the averaged sequence. This variance can be controlled through appropriate adjustment of client learning rates.

Algorithm 1 Federated Averaging Algorithm

Input: Number of global iterations T
Input: Number of local training epoches E
Input: Desired level A_T of model accuracy
Output: Final parameters θ_t
procedure FEDAVG
    ServerGlobalUpdate:
    Initialize parameters θ_0
    Repeat
    $C \leftarrow$ Select a subset of clients
    for each client $c \in C$ **do**
    $\theta_t^c \leftarrow$ **CientLocalUpdate**(θ_{t-1})
    end for
    $\theta_t \leftarrow \sum_C \frac{n^c}{n} \theta_t^c$
    Until A_T is attained
    ClientLocalUpdate(θ)
    for local iteration $e = 1$ to E **do**
    for each each batch b in client's split **do**
    $\theta = \theta - \eta \Delta L(b; \theta)$
    end for
    end for
    return local model θ
end procedure

Figure 6: Federated Averaging Algorithm defined in [3]

2.4.4 Adaptive federated averaging

Traditional federated optimization methods, such as FedAvg, face significant challenges in convergence behavior and hyperparameter tuning. These limitations become particularly evident when addressing non-convex problems, heterogeneous datasets, or communication constraints inherent to FL systems. Recent research has focused on adaptive optimization schemes that outperform traditional methods, especially in environments with prevalent data heterogeneity.

Reddi et al.[17] introduced federated versions of adaptive optimizers—ADAGRAD, ADAM, and YOGI—demonstrating improved convergence behavior in scenarios with significant client dataset variations in distribution, size, or quality. Their analysis revealed crucial relationships between client heterogeneity and communication efficiency, highlighting the challenge of achieving model generalization across diverse datasets while maintaining efficient communication.

FL optimization faces two core challenges: limited adaptivity in traditional Stochastic Gradient Descent (SGD)-based updates and substantial communication overhead from frequent client-server synchronization. Advanced solutions include gradient compression and quantization techniques to reduce communication costs through compressed gradient transmission. Additionally, federated adaptive optimizers like FedAdam enhance model update efficiency by dynamically adjusting learning rates and implementing momentum-based strategies, effectively mitigating client variability impacts and improving convergence rates.

Overall, the development of adaptive optimization techniques represents a significant advancement in addressing the unique challenges of FL, paving the way for more scalable, efficient, and robust federated systems.y

Algorithm 2 Adaptive FL Algorithm

Input:Dataset, privacy budget ϵ, learning rate η
Output:Ψ.
Begin set $\rho_0^1 = 0, \rho_0^2 = 0$.
  For each round of iterations **do:**
    Gradient descent at time step t: $\beta_t \leftarrow \Delta_\Psi f_t(\Psi_{t-1})$
    Calculate the first-order mom: $\rho_t^1 \leftarrow x_1^t \rho_{t-1}^1 + (1 - x_1^t)\beta_t$
    Calculate the second-order mom: $\rho_t^2 \leftarrow x_2 \rho_{t-1}^2 + (1 - x_2)\beta_t^2$
    Clip learning rates by Clip: $(a/\sqrt{(\rho_t^2, \eta_l, \eta_u)}$
    Update the first-order mom estimation:
    $\bar{\rho}_t^1 = \frac{\rho_t^1}{1-x_1^t}$
    Update the second-order mom estimation:
    $\bar{\rho}_t^2 = \frac{\rho_t^2}{1-x_2^t}$
    Until the model converges for the local dataset
  Send the model parameters Ψ_t, of this iteration to the central server
  Central server calculates the contribution $\Psi^i - \Psi^{i-1}$ of the current iteration and delivers it
  End.
Return resulting parameters $\Psi_1^i, \Psi_2^i, \cdots, \Psi_k^i$ from clients to the central server.

Figure 7: Adaptative FL Algorithm[1]

2.4.5 FedProx

As previously discussed, devices in federated networks often have varying resource constraints in terms of computing hardware, network connections, and battery levels. While FedAvg requires each device to perform a uniform amount of work (running the same number of local epochs, E), this approach proves unrealistic in heterogeneous environments. FedProx addresses this limitation by allowing variable amounts of work across devices based on their available system resources, and importantly, aggregates partial solutions from slower devices (stragglers) rather than excluding them.

FedProx accommodates varying local computation capabilities across devices and training iterations, rather than assuming uniform work distribution. While this flexibility helps mitigate system heterogeneity impacts, excessive local updates may still cause

divergence due to underlying data heterogeneity. To address this, FedProx introduces a proximal term to the local subproblem, effectively constraining the impact of variable local updates. This proximal term offers two key benefits: (1) it addresses statistical heterogeneity by keeping local updates closer to the initial global model without manual epoch number configuration, and (2) it safely incorporates varying amounts of local computation resulting from system heterogeneity. Figure 8 summarizes the steps of this enhanced algorithm.

Algorithm 2 `FedProx` **(Proposed Framework)**

Input: $K, T, \mu, \gamma, w^0, N, p_k, k = 1, \cdots, N$
for $t = 0, \cdots, T-1$ **do**
Server selects a subset S_t of K devices at random (each device k is chosen with probability p_k)
Server sends w^t to all chosen devices
Each chosen device $k \in S_t$ finds a w_k^{t+1} which is a γ_k^t-inexact minimizer of: $w_k^{t+1} \approx \arg\min_w h_k(w; w^t) = F_k(w) + \frac{\mu}{2}\|w - w^t\|^2$
Each device $k \in S_t$ sends w_k^{t+1} back to the server
Server aggregates the w's as $w^{t+1} = \frac{1}{K}\sum_{k \in S_t} w_k^{t+1}$
end for

Figure 8: Proposed Framework for FedProx [4].

2.4.6 FedPAQ

FedPAQ, proposed by Reisizadeh et al.[23], introduces a communication-oriented aggregation technique through periodic averaging of local model updates. Unlike traditional approaches where clients synchronize with the server at each iteration, FedPAQ enables multiple local updates before sharing with the server. To reduce communication overhead, the algorithm implements partial device participation, selecting only a subgroup of client devices based on specific criteria: connection to a free wireless network, idle status, and base station reachability.

Similar to FedAvg, FedPAQ computes the new global model by averaging local models, though this approach entails high complexity in both strongly convex and non-convex settings. The main algorithmic steps and implementation details are illustrated in Figure 9.

2.4.7 Turbo-Aggregate

Turbo-Aggregate, proposed by So et al.[5], introduces a communication and security-oriented aggregation technique based on a multi-group circular strategy. The algorithm divides clients into multiple groups, where at each iteration, clients in one group transmit two types of updates: the aggregated model updates from all previous groups and the local model updates from the current group to the next group's clients.

Algorithm 1 FedPAQ

for $k = 0, 1, \cdots, K-1$ **do**
  server picks r nodes $\mathcal{S}_k$ uniformly at random
  server sends $\mathbf{x}_k$ to nodes in $\mathcal{S}_k$
  for node $i \in \mathcal{S}_k$ **do**
    $\mathbf{x}_{k,0}^{(i)} \leftarrow \mathbf{x}_k$
    for $t = 0, 1, \cdots, \tau - 1$ **do**
      compute stochastic gradient
      $\widetilde{\nabla} f_i(\mathbf{x}) = \nabla \ell(\mathbf{x}, \xi)$ for a $\xi \in \mathcal{P}^i$
      set $\mathbf{x}_{k,t+1}^{(i)} \leftarrow \mathbf{x}_{k,t}^{(i)} - \eta_{k,t} \widetilde{\nabla} f_i(\mathbf{x}_{k,t}^{(i)})$
    end for
    send $Q(\mathbf{x}_{k,\tau}^{(i)} - \mathbf{x}_k)$ to the server
  end for
  server finds $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \frac{1}{r} \sum_{i \in \mathcal{S}_k} Q(\mathbf{x}_{k,\tau}^{(i)} - \mathbf{x}_k)$
end for

Figure 9: Proposed Algorithm for FedPAQ [4].

The algorithm incorporates a security component using an additive secret sharing mechanism, adding controlled randomness to each local model to preserve client data privacy. This introduced randomness is designed to neutralize during the local model aggregation process. While Turbo-Aggregate proves particularly suitable for wireless topologies with rapidly changing network conditions and user availability, its secure aggregation mechanism reveals limitations. Although effective in handling user dropouts, the system cannot adapt to new network participants. Future development opportunities include creating a self-configurable protocol that accommodates new users dynamically by reconfiguring system specifications (multi-group structure and coding setup) while maintaining resilience and privacy guarantees. Figure 10 illustrates the detailed algorithm implementation.

Algorithm 1 Turbo-Aggregate

input Local models $\mathbf{x}_i^{(l)}$ of users $i \in [N_l]$ in group $l \in [L]$.
output Aggregated model $\sum_{l \in [L]} \sum_{i \in \mathcal{U}_l} \mathbf{x}_i^{(l)}$.

for group $l = 1, \ldots, L$ **do**
  for user $i = 1, \ldots, N_l$ **do**
    Compute the masked model $\{\widetilde{\mathbf{x}}_{i,j}^{(l)}\}_{l \in [N_{l+1}]}$ from (5).
    Generate the encoded model $\{\bar{\mathbf{x}}_{i,j}^{(l)}\}_{j \in [N_{l+1}]}$ from (10).
    if $l = 1$ **then**
      Initialize $\widetilde{\mathbf{s}}_i^{(1)} = \bar{\mathbf{s}}_i^{(1)} = \mathbf{0}$.
    else
      Reconstruct the missing values in $\{\widetilde{\mathbf{s}}_k^{(l-1)}\}_{k \in [N_{l-1}]}$ due to the dropped users in group $l-1$.
      Update the aggregate value $\widetilde{\mathbf{s}}_i^{(l)}$ from (6).
      Compute the coded aggregate value $\bar{\mathbf{s}}_i^{(l)}$ from (11).
    Send $\{\widetilde{\mathbf{x}}_{i,j}^{(l)}, \bar{\mathbf{x}}_{i,j}^{(l)}, \widetilde{\mathbf{s}}_i^{(l)}, \bar{\mathbf{s}}_i^{(l)}\}$ to user $j \in [N_{l+1}]$ in group $l+1$ ($j \in [N_{final}]$ if $l = L$).
for user $i = 1, \ldots, N_{final}$ **do**
  Reconstruct the missing values in $\{\widetilde{\mathbf{s}}_k^{(L)}\}_{k \in [N_L]}$ due to the dropped users in group L.
  Compute $\widetilde{\mathbf{s}}_i^{(final)}$ from (14) and $\bar{\mathbf{s}}_i^{(final)}$ from (15).
  Send $\{\widetilde{\mathbf{s}}_i^{(final)}, \bar{\mathbf{s}}_i^{(final)}\}$ to the server.
Server computes the final aggregated model from (16).

Figure 10: Proposed Algorithm for Turbo-Aggregate [5].

2.4.8 FedMA

FedMA was proposed by Wang et al.[24] as statistics-oriented aggregation technique that takes into account the permutation invariance of the neurons in the neural network model before performing the aggregation. The objective is to enable global model size adaptation. FedMA employs a Bayesian non-parametric mechanism which allows it adjust the size of the central model to the heterogeneity of data distribution. However, FedMA can be vulnerable to model poisoning attack, where an adversary can easily trick the system to expand the global model in order to accommodate any poisoned local model.

2.4.9 FedMOON

FedMOON (Model-Contrastive Federated Learning) is an innovative approach to federated learning that addresses the challenge of data heterogeneity (non-IID data) across different clients. Developed by Li, He, and Song, FedMOON utilizes model-level contrastive learning to improve performance. Key features of FedMOON:

- Uses model-level contrastive learning (rather than between images) Corrects local training by maximizing agreement between representations learned by the current local model and the global model
- Includes a contrastive loss function that pulls the local model representations closer to the global model while pushing away from previous local model representations
- Employs an architecture with a base encoder, projection head, and output layer

Experiments show that FedMOON significantly outperforms other state-of-the-art algorithms on image classification tasks like CIFAR-10, CIFAR-100, and Tiny-ImageNet, with accuracy gains of up to 7% compared to traditional FedAvg. The algorithm is particularly effective in scenarios with a large number of clients and highly heterogeneous data distributions.

2.4.10 Comparison of algorithms

Comparative evaluations of FL strategies in recent literature demonstrate varying performance across different scenarios. Gao et al.[25] analyzed algorithm performance with heterogeneous datasets, revealing that FedAvg required the most rounds to achieve target accuracy. While FedProx sometimes needed similar or higher round numbers compared to FedAvg, it achieved faster training completion in specific cases. Scaffold demonstrated superior performance by completing training in fewer, faster rounds while maintaining accuracy. Notably, FedDC, combining FedProx and Scaffold approaches, showed exceptional performance across both IID and non-IID datasets. Nguyen et al.[26]

used the CIFAR-10 dataset to evaluate multiple strategies including FedAvg, FedProx, Scaffold, FedNova, FedAdam, FedAdagrad, and FedYogi. Their analysis revealed that ProxYogi, a combination of FedProx and FedYogi, achieved optimal results on non-IID data distributions. In medical applications, Dang et al.[27] compared multiple algorithms for predicting in-hospital mortality and acute kidney injury. FedProx showed the lowest accuracy in acute kidney injury prediction, while most methods achieved high accuracy in mortality prediction, with FedAdam showing slightly reduced performance. Significantly, traditional approaches like FedAvg and FedAvgM demonstrated superior effectiveness for general machine learning tasks, with FedAvg exhibiting particular strength with IID datasets. These comparative studies highlight that algorithm selection should consider specific use case requirements, data distribution characteristics, and computational constraints to achieve optimal performance.

The literature demonstrates that each FL algorithm offers unique advantages depending on the specific use case. A summary of these advantages for commonly utilized strategies is presented in Table 1.

Table 1: Comparison of FL Algorithms [2]

FL Algorithm	**Utility**
FedAvg	Simple and common
FedAdam	Effective optimization
FedYogi	More stable learning
FedAdagrad	Automatic learning rates
FedPAQ	partial device participation
FedProx	Better for non-IID data
Scaffold	More stable convergence
FedMA	employs Bayesian non-parametric
FedMoon	model-contrastive learning

2.5 Datasets

There are several standardized datasets that are widely used and have become highly competitive in many field of research. In this section we describe some datasets that are frequently used in FL problems. A single dataset may only cover a specific task, because of this is important to evaluate a varied suite of benchmark tasks, allowing a more holistic approach to assessing and characterizing the performance of an algorithm or system.

2.5.1 MNIST Dataset

The MNIST database (Modified National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems. The database is also widely used for training and testing

in the field of machine learning. It was created by "re-mixing" the samples from NIST's original datasets. The creators felt that since NIST's training dataset was taken from American Census Bureau employees, while the testing dataset was taken from American high school students, it was not well-suited for machine learning experiments. Furthermore, the black and white images from NIST were normalized to fit into a 28x28 pixel bounding box and anti-aliased, which introduced grayscale levels.

The MNIST database contains 60,000 training images and 10,000 testing images. Half of the training set and half of the test set were taken from NIST's training dataset, while the other half of the training set and the other half of the test set were taken from NIST's testing dataset.

Extended MNIST (EMNIST) [28] is a newer dataset developed and released by NIST to be the (final) successor to MNIST. MNIST included images only of handwritten digits. EMNIST includes all the images from NIST Special Database 19, which is a large database of handwritten uppercase and lower case letters as well as digits. The images in EMNIST were converted into the same 28x28 pixel format, by the same process, as were the MNIST images. Accordingly, tools which work with the older, smaller, MNIST dataset will likely work unmodified with EMNIST.

2.5.2 EMNIST Dataset

The Extended MNIST (EMNIST) dataset represents an important benchmark for evaluating federated learning algorithms. Derived from the NIST Special Database 19, EMNIST extends the original MNIST dataset by including handwritten letters in addition to digits. The dataset contains 814,255 characters from 62 unbalanced classes (digits 0-9, lowercase and uppercase letters a-z, A-Z), making it significantly larger and more diverse than MNIST. In federated learning research, EMNIST is often partitioned to simulate realistic non-IID scenarios where clients possess data from different subsets of classes or with varying class distributions. Several studies, including those by SCAFFOLD and FedProx, have utilized EMNIST to evaluate the performance of federated optimization methods when dealing with heterogeneous data distributions. The dataset's characteristics make it particularly useful for testing algorithm robustness in addressing client drift and convergence challenges in federated environments.

2.5.3 Stack Overflow Dataset

Stack Overflow is a language dataset consisting of question and answers from the Stack Overflow online forum. It is available in TensorFlow Library [29]. It has questions and answers have associated metadata, including tags (e.g., "), the creation time of the post, the title of the associated question, the score of the question, and the type of post (i.e., whether it is a question or an answer). Each client corresponds to a user, whose examples are all of their posts. The version available in [29] partitions the dataset among training, validation, and test clients. There are 342,477 train clients, 38,758 validation

clients, and 204,088 test clients. Notably, the train clients only have examples from before 2018-01-01 UTC, while the test clients only have examples from after 2018-01-01 UTC. The validation clients have examples with no date restrictions, and all validation examples are held-out from both the test and train sets. The dataset is relatively unbalanced, with some users having very few posts, and other having many more (one client has over 80,000 examples).

2.5.4 CIFAR-10 dataset

The CIFAR-10 dataset was proposed by Alex Krizhevsky and Geofrey Hinton[30]. It is a computer vision dataset consisting of 32x32x3 images with 10 possible labels. There are 50,000 training examples and 10,000 test examples, for a total of 6000 propose a technique for creating non-IID partitions of the dataset across clients. Each client draws a multinomial distribution over the 10 labels from an underlying symmetric Dirichlet distribution. To assign the client examples, we draw labels from this multinomial distribution, and sample (without replacement) one of the images in CIFAR-10 with the corresponding label. Usually, the entire dataset is used to distributed data among clients or devices.

2.5.5 GLD-23k and GLD-160k Datasets

Hsu et al. proposed the GLD-160k dataset[31](under the name Landmarks-User-160k). It is a federated version of the centralized Google Landmark Dataset. Each client represents an author, and the images are partitioned into clients based on their original authorship. GLD-160k contains 164,172 training images, 2028 landmark labels, and 1,262 clients. It has a single test set (i.e., the test dataset is not federated) containing 19,526 images. The GLD-23k dataset is a subset of the GLD-160k dataset. GLD-23k contains 23,080 training images, 203 landmark labels, and 233 clients. Same as GLD-160k, GLD-23k has a single test set containing 1,959 test examples, which covers all the labels in the federated training set (see the rightmost plot of Figure 10). Though smaller, the GLD-23k dataset maintains the imbalanced training characteristics of the GLD-160k dataset [31]. The number of training examples per label and the number of training examples per client both can vary for an order of magnitude, and the number of training labels in each client can vary for up to two orders of magnitude. These characteristics make the GLD-23k dataset representative of some issues endemic to real-world FL problems, and the smaller size makes it easier to work with in simulations.

2.5.6 Shakespeare Dataset

Shakespeare is a language modeling dataset derived from The Complete Works of William Shakespeare. It consists of 715 characters whose contiguous lines are examples in the client dataset. The train set has 16,068 examples and test set has 2,356 examples. The preprocessing for EMNIST and Shakespeare are provided by the Leaf project [32],

which also provides federated versions of the sentiment140 and celebA datasets. These datasets have enough clients that they can be used to simulate cross-device FL scenarios, but for questions where scale is particularly important, they may be too small. In this respect Stackoverflow provides the most realistic example of a cross-device FL problem.

Luo et al.[33] introduced a federated computer vision dataset comprising over 900 annotated street images from 26 street cameras, featuring seven object categories with detailed bounding box annotations. However, the relatively small sample size may not adequately represent challenging real-world scenarios. The development of comprehensive FL datasets that reflect real-world problems remains a crucial challenge for the research community. While platforms like TensorFlow Federated encourage and support the contribution of new datasets, researchers can also create federated scenarios by strategically partitioning existing open datasets into client-specific splits. These partitions often benefit from unbalanced and non-IID distributions to better simulate real-world conditions. Preserving auxiliary information such as timestamps, geolocation, and other metadata proves valuable for comprehensive analysis and evaluation of FL systems.

2.6 Strategies to improve algorithm performance

One of the main challenges in FL arises when feature distributions vary significantly across clients. Different clients typically have distinct interests and activities, leading to considerable variation in local dataset features. For example, in a federated training task predicting human face attractiveness investigated by Wahab et al.[3], clients may have varying cultural or demographic preferences that influence their data distributions.

This feature variation extends beyond simple preferences. Clients may fundamentally differ in how they categorize or label similar items. While one group might classify certain features as 'critical,' another might deem them 'non-essential,' making it challenging for a single model to accurately predict across all client populations.

The technical impact of distribution imbalance becomes particularly significant given FL's reliance on stochastic gradient descent (SGD) for training deep neural networks. In federated settings, the limited size of local datasets means that stochastic gradients often closely approximate the full gradient from each client's data, thereby amplifying the effects of data heterogeneity. To address these data distribution challenges, four primary techniques have emerged: multi-task learning, client clustering, transfer learning, and parameter tuning.

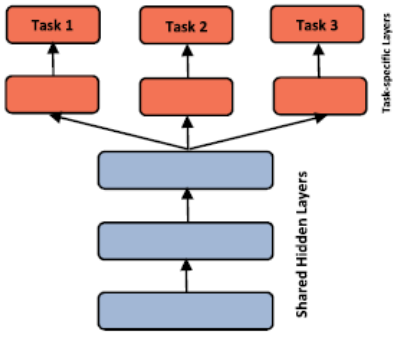
2.6.1 Multi-Task Learning

Traditional machine learning typically focuses on optimizing a single metric through training and fine-tuning a model or ensemble of models. While this approach has demonstrated success across various domains, limiting focus to a single task may overlook valu-

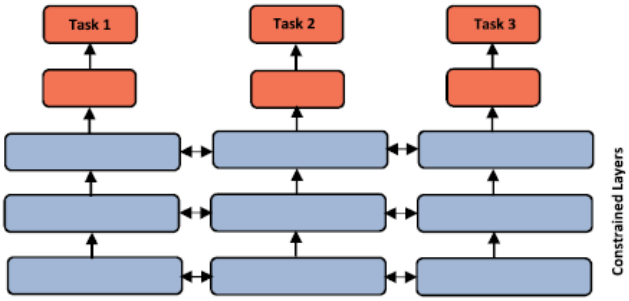
able information that could enhance the original metric's optimization. Training multiple related tasks with shared representations often enables machine learning models to achieve better generalization on their primary task.

This approach, known as Multi-Task Learning (MTL), can be illustrated through an analogy to research training. When preparing students for research careers, the curriculum extends beyond scientific methodologies to include writing, communication, and presentation skills. Although seemingly distinct, these complementary skills enhance overall research capability. MTL implementation primarily uses two methods: hard parameter sharing and soft parameter sharing. Hard parameter sharing, primarily used to reduce overfitting, maintains shared hidden layers across all tasks while preserving task-specific output layers, as illustrated in Figure 11. Conversely, soft parameter sharing assigns separate models with individual parameters to each task.

In FL, soft parameter sharing MTL has emerged as an effective approach to address non-IID data challenges by assigning distinct task models to different clients. This approach capitalizes on the inherent structure among client task models—for example, similar user behaviors in mobile phone usage—to capture relationships among non-IID data through various relatedness metrics (graph-based relationships, sparsity patterns, etc.).



(a) Multi-Task Learning with Hard Parameter Sharing Method



(b) Multi-Task Learning with Soft Parameter Sharing Method

Figure 11: Multi-Task Learning: Several related tasks are trained together and the representations are sharing among them [3].

2.6.2 Client Clustering

The client clustering approach aims to group clients into clusters that share common characteristics. The ultimate goal of such an approach is to facilitate the adoption of MTL in the FL framework. Specifically, the objective of federated MTL is to assign each client with a model that optimally fits its local data distribution. This, however, contradicts with the FL vision wherein all clients are treated equally and only one single global model is learned. To bridge this gap, the objective of client clustering approaches is to group clients into disjoint groups and hence assign a single task model to each cluster.

The clustering is achieved through assessing the similarity among the clients' gradient updates. The main intuition is that clients with similar gradients tend to have similar data distributions. Consequently, clients that belong to the same cluster can jointly train a single model, which can be considered one task of the whole federated MTL.

For example, Sattler et al.[34] depart from the observation that FL yields suboptimal results in case where the data distributions of the clients' local data are divergent to propose a clustering strategy that takes advantage of geometric properties of the FL loss surface to cluster the clients into groups with jointly trainable data distributions. The idea is that the cosine similarity among the weight updates of the different clients is a hint about the similarity in their data distributions. Briggs et al.[35] proposed to group the clients based on the similarity of their local updates to the global joint model using a hierarchical clustering mechanism. The clusters are independently trained in a parallel fashion on specialized machine learning models.

2.6.3 Transfer Learning

Transfer learning is a machine learning approach that, instead of initiating the training process from scratch, gives the chance to start from patterns that were already learned through solving a different problem. For example, if you trained a simple classifier to predict whether an image contains a table, you could use the knowledge that the model gained during its training to recognize couches[3].

Transfer learning addresses the problem of performance degradation in supervised learning when we modify the domain or task but do not have enough labeled data for that new domain or task. For instance, suppose that we trained a model to detect animals on images taken during the daytime. In this scenario, the task is detecting animals and the domain is that of daytime. Now, if we want to train another model to detect animals in images taken during the night-time (a different domain), we would need millions of labeled images taken during the night-time. The performance of the model trained on daytime images would considerably degrade when applied to nighttime images since this model does not know how to generalize to a new domain, i.e., night-time.

In general, the transfer learning vision is achieved through employing a pre-trained model, i.e., a model that has been trained on a large dataset to solve a problem that is similar to the one you want to solve. We first train a base source network on a base dataset and a base task. Then we re-purpose the learned features (or transfer them) to a second target network to be trained on a target dataset and task. Basically, we transfer the weights that the model has learned from "task A" to a new "task B". This process is beneficial provided that the features are general, i.e., appropriate for both the base and target tasks, rather than being specific to the base task. Technically speaking, transfer learning benefits from the fact that the features in the lower levels of the deep network are highly transferable. This is because these features focus on learning common and low-level features. Hence, the parameters that are obtained from the source model can be transferred to the target model so as to learn the personalized features of that model.

The general idea of transfer learning is schematized in Fig. 12.

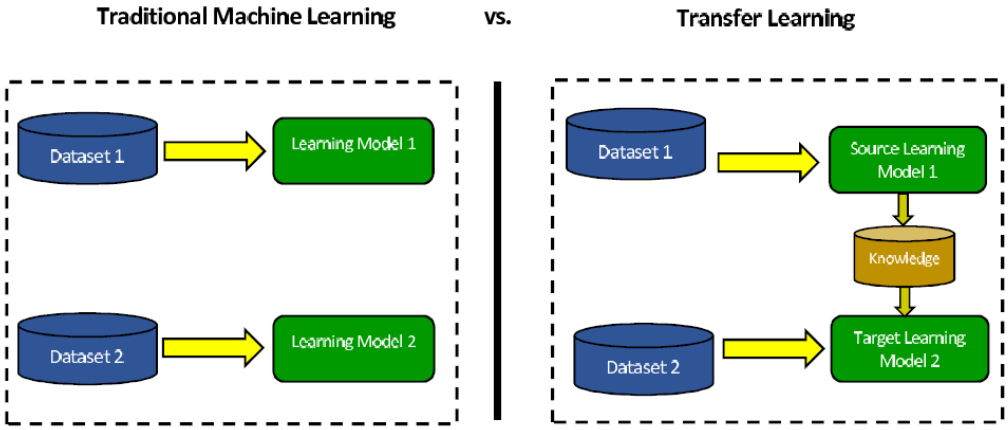


Figure 12: Transfer Learning: The machine learning model capitalizes on some patterns that were already learned through solving a previous problem [3].

Transfer learning is used in the context of FL to address the limited applicability problem of FL to only vertically or horizontally partitioned data. Given that most existing datasets have a small number of common features and/or population and a majority of non overlapping data [36], most of the data would be under-utilized by the FL framework. Transfer learning is proposed as a potential solution to such a challenge as this approach does not impose any limitation on the distribution of the data and can hence provide results for the entire population and feature space. For example, Liu et al.[37] proposed a privacy-preserving federated transfer learning approach to deal with the heterogeneity in the clients' data distributions. Figure 12 illustrated the transfer learning.

2.6.4 Parameter Tuning

Parameter tuning approaches aim to improve federated training convergence on non-IID data by modifying stochastic gradient descent methods. Two primary strategies emerge: momentum and parameter correction. The momentum strategy focuses on optimizing gradient vector directions through weighted averaging methods. Momentum functions as a denoising technique, using moving averages to approximate the original function (whether exponential, linear, or other forms). In FL, SGD with momentum averages gradients received from clients across iterations to optimize gradient vector direction. Liu, Wei et al.[38] introduced Momentum Federated Learning (MFL), which incorporates the momentum term during local updates. Unlike traditional gradient descent that considers only current gradients, MFL integrates both the previous iteration and current gradient to accelerate convergence. The server performs weighted averaging of received local parameters after each iteration to derive both global momentum and model parameters before transmission back to clients. Further developments were presented by Jiang et al.[39] with a consensus-based distributed momentum SGD model, addressing three key requirements: decentralized computation, data parallelization, and constrained communication.

Parameter correction represents the second major strategy in parameter tuning techniques, focusing on adjusting specific parameters to achieve targeted improvements. Koskela et al.[40] developed an approach that corrects learning rates to optimize the global model for each client's local data at training iteration starts. Complementarily, Li et al. [4] introduced corrections to client local subproblems to constrain local updates and maintain closer alignment with the initial global model.

The motivation for learning rate adaptation stems from a fundamental challenge: after averaging client local models, the global model often deviates from optimal performance on individual client datasets, necessitating adjustments at each iteration's beginning. Building on this understanding, Li et al.[4] developed FedProx as an enhancement to FedAvg, specifically addressing statistical heterogeneity arising from non-identical data distribution across devices. FedProx introduces a proximal term to the local subproblem on each client's device, effectively constraining local update influence while accommodating variable computational workloads across devices.

Wang et al.[24] addressed a key limitation in FedAvg: performance degradation of the averaged global model caused by coordinate-wise weight averaging. This issue arises from neural network parameter permutation invariance, where multiple network variants exist with identical functionality but different parameter orderings. Their solution, Federated Matched Averaging (FedMA), implements a layer-wise federated learning approach that accounts for neuron permutation invariance before aggregation, enabling dynamic adaptation of global model size. Mostafa et al.[41] developed two complementary strategies to address data heterogeneity. The first implements adaptive online hyperparameter tuning, while the second introduces a regularization term to penalize divergent representation learning across clients. Their approach frames hyperparameter selection as an online reinforcement learning problem: learners select hyperparameter values as actions during each round, receiving rewards based on training loss reduction at round completion.

2.7 Client selection

Client selection is a fundamental component of FL that determines participant clients in each training round. The process aims to identify clients that can provide valuable updates to the global model while balancing data quality, resource constraints, and communication efficiency. The selection process ranges from random sampling to sophisticated approaches that consider data distribution, resource availability, and client performance history [42] . The choice of selection strategy significantly impacts FL performance-effective selection can enhance model convergence and generalization, while poor selection may lead to slower learning, bias, and potential security risks. Recent advances in client selection research have introduced techniques based on reinforcement learning [42], multi-armed bandits, and game theory. These approaches optimize selection based on data diversity, resource utilization, and fairness. However, the field still faces

significant challenges due to heterogeneous client characteristics in terms of availability, data distribution, and system capabilities [4].

Client selection is a crucial component of FL that directly impacts the performance, efficiency, and fairness of the learning process. However, selecting the optimal set of clients to participate in each round of FL poses several challenges. In this section, we discuss the key challenges in client selection, such as data heterogeneity, concept drift, imbalanced data, resource constraints, heterogeneous hardware, and others.

One of the primary challenges in client selection stems from non-IID (non-independent and identically distributed) data distributions across clients. In real-world FL scenarios, clients generate data with significantly different distributions, leading to diverse local model updates. This heterogeneity creates a critical challenge: selecting clients with highly skewed or divergent data distributions may impede global model convergence and degrade overall system performance. Effective client selection strategies must therefore balance data heterogeneity by selecting clients with representative, diverse, and high-quality data that contribute meaningfully to the learning objectives.

Concept drift presents another significant statistical challenge for client selection, occurring when underlying data distributions evolve over time. In dynamic FL environments, client data continuously evolves due to changing user behaviors, seasonal patterns, or external factors. These temporal changes can significantly impact the effectiveness of static client selection strategies. To maintain model relevance and performance, selection methods must dynamically adapt to these distribution shifts. Approaches such as adaptive importance sampling and online learning have proven effective in addressing concept drift by continuously updating selection criteria based on the evolving data landscape.

Imbalanced data distribution among clients significantly impacts FL performance. When clients possess vastly different dataset sizes, selection based solely on data volume can lead to biased model updates and underrepresentation of minority classes. To address this, effective client selection strategies must balance sufficient data quantity with fair class representation. Mitigation techniques include stratified sampling, data augmentation, oversampling, undersampling, and class-weighted loss functions, all promoting more equitable contributions to the global model.

Client devices in FL operate under various constraints, including limited computational power, memory, storage, and battery life. This hardware heterogeneity leads to differences in training time and model update quality, potentially impacting fairness and workload distribution. Client selection strategies must therefore consider both resource availability and device capabilities to ensure balanced participation without causing performance degradation or excessive resource consumption. Solutions include resource-aware selection, device-aware selection, model compression, and quantization techniques.

Communication challenges in FL manifest through bandwidth limitations and latency issues, particularly in wireless or low-bandwidth networks. These constraints can affect selection efficiency and slow global model convergence. To address these challenges, client selection strategies employ various approaches: Bandwidth-aware selection and

model compression for efficient data transfer; Latency-aware selection and asynchronous communication for timely updates; Client clustering and hierarchical federated learning for improved network efficiency; Edge computing with local aggregation for reduced communication overhead.

Selection bias occurs when chosen clients fail to adequately represent the overall data distribution, potentially leading to unfair treatment of certain groups. Ensuring both fairness and representativeness in client selection is crucial for model generalization and robustness. Effective strategies employ diversity-aware selection, fairness constraints, and stratified sampling to capture population diversity while maintaining equal participation opportunities.

Client failures and dropouts frequently occur in FL environments due to network disconnections, battery depletion, or user interruptions. These disruptions can significantly impact client selection processes and degrade overall system performance. To enhance fault tolerance, FL systems employ multiple strategies: Backup client selection and redundancy mechanisms, Asynchronous client selection protocols, and Fault-tolerant aggregation algorithms.

Security concerns present additional challenges, particularly regarding malicious clients who may manipulate local data or model updates to compromise global model integrity. Protective measures include: Anomaly detection systems, Reputation-based client selection, Secure aggregation protocols, and Privacy-preserving communication techniques.

These approaches work together to safeguard against data leakage and model poisoning attacks while maintaining system reliability. Client selection in FL must address multiple interconnected challenges across statistical, systemic, communicational, and security domains. Successful implementation requires sophisticated selection strategies that adapt to FL systems' dynamic and heterogeneous nature while ensuring learning process efficiency, robustness, and fairness. The following chapter explores advanced client selection techniques designed to address these challenges and enhance FL system performance and reliability.

2.8 Chapter Conclusions

This chapter provided some of the main concepts regarding our thesis proposal. Given some of algorithms, datasets most used in FL problems and also some challenges related to client selection. In this thesis, We choose the fine tuning parameter and client selection strategies to be further investigated in the following chapters.

CHAPTER 3

Related Works

This chapter presents the state-of-the-art in FL, examining optimization strategies for local updates and client selection algorithms across diverse FL system applications. While our research builds upon established approaches, it extends current methodologies by addressing emerging challenges in FL implementation.

3.1 Centralized to Decentralized Learning

The evolution from centralized cloud-based to distributed edge-server learning has given rise to FL. This paradigm enables local data retention and model training on devices and servers, preserving data privacy and confidential information while addressing storage and communication costs. The key advantages of FL include offline operability, improved latency and accuracy, enhanced privacy protection, extended battery life, and localized model training.

FL's decentralized approach minimizes data transfer overhead by keeping raw data on devices and performing local ML training. The accumulated knowledge is then aggregated and shared among participants through model federation on a central server. Abdulrahman et al.[8] presented a novel classification of FL challenges and research domains based on technical barriers and ongoing developments. Similarly, Khan et al.[43] outlined recent FL advancements enabling IoT applications, establishing comprehensive metrics including sparsification, resilience, quantization, scalability, security, and privacy.

Traditional centralized cloud-based learning faces increasing challenges due to transmission and storage costs, alongside privacy concerns. FL addresses these issues by enabling collaborative training of ML models across multiple clients without centralized data transfer. However, implementing FL in IoT networks still faces challenges across var-

ious sectors, including manufacturing, transportation, energy, healthcare, and computing. Zhang et al.[44] discussed challenges and potential solutions for IoT-based FL systems.

In the following subsection, we summarize some related research that addressed the problem of communication efficiency in FL. Many of than explored alternative algorithms to increase the accuracy and reduce the loss considering the heterogeneity.

3.2 Algorithms for communication efficiency

Several works have been proposed to improve the FL. Some of them propose customization of both server and client. Li et al.[4] proposed FedProx, including a component to represent the statistical heterogeneity across client devices. The authors added a proximal term to the local training subproblem on each client device. This adaptation allows the local updates to get closer to the initial global model, limiting the influence of each local model update on the global model. This adaptation is proposed to improve the convergence when dealing with statistically heterogeneous data, and when all the devices are weighted equally in the global aggregation phase, disregarding the differences in the device capabilities (e.g., hardware).

Reisizadeh et al.[23] proposed FedPAQ focusing on periodic averaging of the local model updates. Instead of synchronizing their model updates with the server at each iteration, it enables clients to carry out multiple local updates on the model prior to sharing the updates with the server. The clients should be chosen to participate in the training at each iteration on the basis of factors such as the device connectivity to a free wireless network, idle, and reachable to a base station. No evaluation of overfitting or fairness was performed in this work.

So et al.[5] introduced a communication and security-oriented aggregation technique based on a multi-group circular strategy. Turbo-Aggregate divided the clients into multiple groups and at each iteration the clients belonging to one group transmit the aggregated model updates of all clients in the previous groups and the local model updates of the current group to the clients of the next group. It includes a security component for sharing mechanisms and preserve the privacy of clients' data. This proposition does not adapt to new users that join the network and could be extended to a self-configurable protocol.

In addition, Wang et al.[24] proposed the FedMA, a statistics-oriented aggregation technique which uses permutation invariance of the neurons in the neural network model before performing aggregation. It aims to adapt the model size employing a Bayesian non-parametric mechanism which allows it to adjust the size of the central model to the heterogeneity of data distribution. Unfortunately, FedMA can be vulnerable to model poisoning attacks and some additional mechanisms should be added to improve security.

The problem of 'client-drift' is investigated by Karimireddy et al.[18], when data are heterogeneous (non-iid), resulting in unstable and slow convergence. The authors pro-

posed a new algorithm called Stochastic Controlled Averaging algorithm (SCAFFOLD), that uses control variate to reduces the effect of data heterogeneity and client sampling. This work also indicates that most of the methods benefit from an increasing number of epochs but they didn't evaluate potential overfitting.

The non-IID data distribution in FL emerges from inherent heterogeneity in local data generated across client devices. As each device records its owner's unique activities, data across devices exhibits varying sizes, features, and target class distributions. Consequently, local data from any single client cannot adequately represent the overall data distribution[45]. While federated optimization methods using local updates can significantly reduce required communication rounds for convergence, data heterogeneity may lead to slower convergence, reduced stability, or even divergence.

Our experimental work evaluates how increasing the number of epochs (local updates) affects FedAVG and FedADAM performance, following the protocol established in by Reddi et al.[17]. Based on our analysis of impact and potential model overfitting, we propose an adaptive controller that varies epoch numbers using a Poisson distribution to mitigate overfitting risks. Building on the previously discussed FedAVG variations, our research implements several algorithm variants, recognizing that different approaches achieve optimal results in specific scenarios. Given FL's diverse application domains, a flexible approach to algorithm selection proves advantageous. Table 2 summarizes key findings from related works comparing these algorithms' performance across different contexts.

3.3 Client Selection

Random client selection, the standard method, yields suboptimal results. Several client selection mechanisms have been proposed recently, each of them applied to some specific scenarios, architecture, or application. For instance, Qu et al.[46] proposed the Context-aware Online Client Selection (COCS), which allows clients to use their computational information, bandwidth, and distance to select a subset of clients to maximize the training utilities. Qiao et al.[47] also proposed a context-aware approach, where the central server evaluates the significance of clients based on both the content of their local updates and communication channel states.

Ami et al.[48] introduced the Multi-Armed Bandit (MAB) for client selection, which formulates the trade-off between the training latency and the model's generalization. of the model. MAB reduces the training process time while increasing the model's generalization ability by avoiding overfitting. The authors proposed selecting clients based on a time-varying reward influenced by the history of previous selections.

Sousa et al.[49] introduced an entropy-based client selection for vehicular FL environments to address issues arising from non-IID data distributions. The proposed method is compared to a random selection mechanism in both IID and non-IID scenarios, and scenarios with random client drops.

Table 2: Summary of FL Strategy Comparisons Across Different Studies.

Study	**Dataset**	**FL Strategies**	**Key Findings**
Gao et al.[25]	Heterogeneous datasets	FedAvg, FedProx, Scaffold, FedDC	- FedAvg required the most rounds to converge. - FedProx completed training faster in some cases. - Scaffold converged faster with fewer rounds. - FedDC outperformed others on IID and non-IID datasets.
Nguyen et al.[26]	CIFAR-10 (Non-IID)	FedAvg, FedProx, Scaffold, FedNova, FedAdam, FedAdagrad, FedYogi, ProxYogi	- ProxYogi achieved the best test accuracy. - Other strategies were less effective in non-IID scenarios.
Dang et al.[27]	Medical data (hospital mortality and acute kidney injury)	FedAvg, FedAdam, FedYogi, FedAdagrad, FedProx, FedAvgM	- FedProx had the lowest accuracy for acute kidney injury. - Most strategies achieved high accuracy for hospital mortality. - FedAvg and FedAvgM were the most effective overall, with FedAvg excelling on IID datasets.

Ouyang et al.[50] introduced a cluster indicator matrix indicating the similarity of users called ClusterFL. In this work, the aggregation server drops stragglers which converge slower and clients which are less related to others in each cluster. This work aims to reduce the overall communication overhead while maintaining the overall accuracy performance. Sattler et al.[51] presented a Clustered FL (CFL) that combines privacy constraints by applying an encryption mechanism with clustering. This work uses the geometric structures of a surface to calculate the positioning and grouping of some clients. This method helps to achieve improved performance due to greater equality between the trained client weights and their modes.

Nguyen et al.[52] proposed a fast convergent FL algorithm called FOLB, which performs intelligent sampling of devices in each round to improve the expected convergence speed. The clients are sampled based on the re-weighting mechanism of updated parameters received from participating devices in every round. FOLB is capable of han-

dling the heterogeneity of device communication and computation by using their model to create groups with the ability to achieve faster convergence. This work also evaluates the heterogeneity of devices by only assigning random images from a fixed number of different digits to each device with MNIST Dataset. The authors compared the number of rounds necessary to reach a certain accuracy level to indicate its communication network reduction cost.

Ghosh et al.[53] proposed an Iterative Federated Clustering Algorithm (IFCA), which considers user cluster identity estimation and gradient descent-based model parameter optimization. In this way, IFCA distributes and clusters users, where diverse groups pursue individual learning tasks collaboratively. The clusters may represent groups of users interested in different categories of news. The authors used data augmentation to simulate an environment where the data on different worker machines are generated from different probabilistic distributions.

Albaseer et al.[54] introduced Clustered Federated Multitask Learning (CFL) as an efficient method for developing reliable specialized models in non-IID scenarios. The cluster is designed using the cosine similarity between the weight updates of diverse clients. All clients have an equal probability of participating in the training phase, even if they have wrong channels or low computational capabilities to obtain unbiased models. The client's expected latency is the parameter used to schedule the model uploading, which can be the bottleneck of this approach. Huang et al.[55] propose an active client selection for clustered federated Learning (CFL) to address data heterogeneity by customizing models for different client groups. This work combines many CFL client selection strategies, including calculating the least confidence, margin, entropy, vote entropy, and loss to find the most informative and helpful clients.

Fraboni et al.[56] introduced a clustered sampling for client selection, demonstrating better client representativity and reduced variance in aggregation weights. They introduced two clustering approaches based on sample size and model similarity. The approach seamlessly integrates into FL implementations without client-side operations, remaining compatible with privacy and communication reduction technologies. Li et al.[57] proposed a soft clustering method with the clients being partitioned into overlapping clusters, and the information of each participating client is used by multiple clusters simultaneously during each round. Client selection contributes to improving the model's results and even indirectly contributes to reducing communication since only a portion of the clients participate in the process.

Table 3 summarizes the main characteristics of reviewed studies regarding the challenge of select clients and summarizes the existing works regarding the Clustering approach, Clients Selection Method, and Communication Cost Mitigation.

Based on the state-of-the-art analysis, we argue that it is important to group users based on the statistical features of their datasets to discover the relationships within user datasets, mitigating the impact of non-IID data. In this context, clustering algorithms are an important tool for grouping participants with comparable data distributions or

Table 3: Summary of related works and comparison with our MESFLA mechanism

Papers	**Cluster Approach**	**Selection Method**	**Communication Cost Mitigation**
Qu et al.[46]	X	Computational and client-ES pairs transmission information	Hierarchical FL reduces communication to the cloud server, using nearby edge servers instead
Qiao et al.[47]	X	Client Local Updates and Communication Channel	Selects the client subset based on both the wireless channel states and the content of model updates
Ami et al.[48]	X	Time-varying reward influenced by the history of previous selections	Developed a time-varying reward function that captures the client latency
Sousa et al.[49]	X	Entropy-based for Vehicular FL environments	Focused only in the challenges posed by non-IID data in vehicular networks
Ouyang et al.[50]	✓	Similarity of users and drop stragglers	Reduces overall communication overhead through cluster-wise straggler dropout and correlation-based node selection
Sattler et al.[51]	✓	Similarity with client weights after the convergence	Focused only on properties of the FL loss surface to group the client population into clusters
Nguyen et al.[52]	✓	Re-weighting mechanism of updated parameter	Communication and heterogeneity is improved with the re-weighting mechanism of updated parameters
Ghosh et al.[53]	✓	Similarity based on weight sharing technique	Focused only on convergence of Iterative Federated Clustering Algorithm (IFCA)
Albaseer et al.[54]	✓	Cosine similarity between the weight-updates	Schedule the clients based on their round latency and exploits the bandwidth reuse
Huang et al.[55]	✓	Least confidence, margin, entropy, vote entropy, and loss	Reduces communication overhead requiring less client participation in the learning process
Li et al.[57]	✓	Soft clustering with overlapping clusters	Focused only on combining the strengths of soft clustering and IFCA
MESFLA (Proposed in this thesis)	✓	Data weight and similarity(entropy) cluster with convergence rate monitor	Combines clusterization and metrics evaluation to reduce communication between aggregation server and clients

learning objectives, creating clusters with similar models.

In addition, we could select significant or relevant users in each group, which could be achieved by considering data weight and similarity between trained model representations to mitigate the effects of non-IID data. This approach helps to select the most relevant clients for each training round, reducing the amount of data transmitted and improving overall communication efficiency. In addition, a large number of devices may participate in FL over unreliable networks, leading to significant communication costs and performance bottlenecks. In this way, it is important to only send the new model when accuracy reduces its improvement rate. To the best of our knowledge, only considers every critical characteristic previously mentioned not provided by the existing client

selection mechanism. Our proposal called MESFLA is detailed in Chapter 5.

3.4 Related Works Conclusions

This chapter has presented a comprehensive review of the state-of-the-art in Federated Learning, with particular focus on optimization strategies for local updates and client selection mechanisms. The literature reveals significant challenges in implementing FL systems, especially regarding communication efficiency and data heterogeneity across clients. From our analysis, several key points emerge:

The evolution from centralized cloud-based to distributed edge-server learning has brought advantages in privacy preservation, reduced data transfer overhead, and localized model training, but has introduced new technical challenges. Various algorithms have been proposed to address communication efficiency in FL, including FedProx, FedPAQ, Turbo-Aggregate, FedMA, SCAFFOLD, and FedMOON, each offering unique approaches to improve convergence and handle non-IID data. Client selection mechanisms have progressed beyond random selection, with approaches like COCS, MAB, entropy-based selection, and clustering techniques showing promise for enhancing model performance while reducing communication costs. Clustering approaches, in particular, demonstrate potential for grouping users with similar data distributions or learning objectives, thereby mitigating the impact of data heterogeneity. The reviewed literature highlights that no single approach optimally addresses all challenges in FL implementation. Different algorithms achieve optimal results in specific scenarios, underscoring the need for adaptable approaches to algorithm selection based on application domains and data characteristics. Based on this analysis, our research aims to extend current methodologies by developing MESFLA, a novel approach that combines clustering techniques with metrics evaluation to select relevant clients in each group while monitoring convergence rates to optimize communication efficiency. This approach promises to address the identified gaps in existing client selection mechanisms and contribute to more efficient and effective FL systems.

CHAPTER 4

Local Adaptations to improve communication efficiency

Federated optimization methods that perform local updating can significantly reduce communication rounds needed for convergence. However, heterogeneity can potentially lead to slower convergence, reduced stability, or divergence. In our experiments, we evaluate the impact of the increasing number of epochs (local updates) of FedAVG and FedADAM using the same protocol proposed by Reddi et al.[17]. With the proven impact and possible overfitting of the model, we propose an adaptive controller to vary the number of epochs using a Poisson distribution to avoid overfitting.

4.1 Empirical comparison between algorithms

First, we present an empirical comparison between FedAVG and FedADAM in different scenarios involving different local update settings. The aim is to assess how the number of epochs(local updates) can impact the convergence, especially in cross-device settings. To accomplish this, we conduct simulations employing the dataset EMNIST. It consists of images of digits and upper and lower case English characters, with 62 total classes. The federated version of EMNIST [32] partitions the digits by author. The dataset has natural heterogeneity stemming from the writing style of each person. We train a Convolutional Network for character recognition. The network has two convolutional layers (with 3 x 3 kernels), max pooling, and dropout, followed by a 128 unit dense layer similar as proposed by Reddi et al.[17].

The implementation was based on TensorFlow Federated and we should highlight some features as the clients are sampled uniformly at random, without replacement in a given round, but with replacement across rounds. Second, instead of performing K

training steps per client, we perform E epochs of training over each client's dataset. Last, to account for varying numbers of gradient steps per client, we weight the average of the client outputs by each client's number of training samples. This follows the approach adopted by McMahan et al.[15].

An implementation based on FedOpt was used. For the FedAVG simulations, we used ClientOpt and ServerOpt with SGD with learning rates 0.1 and 1.0 for client and server, respectively; The batch size was defined in 20 for EMNIST. For FedAdam, the client learning rate was 0.03 and the server was at 0.003. Other parameters were defined as proposed by Reddi et al.[17], which showed an extensive evaluation and grid-search of the best parameter values to minimize the average training loss over the last 100 rounds of training.

The simulations also collected some validation metrics. They were measured on a validation set throughout training using the entire test set. The number of communication rounds is the main parameter to evaluate how fast is the convergence, but a more complex and realistic scenario can be defined.

We perform 1500 rounds of training and Figure 13 shows how the increase in local updates can improve the speed of convergence. The metric loss and accuracy improve needing fewer communication rounds as demonstrated in Table 4 .

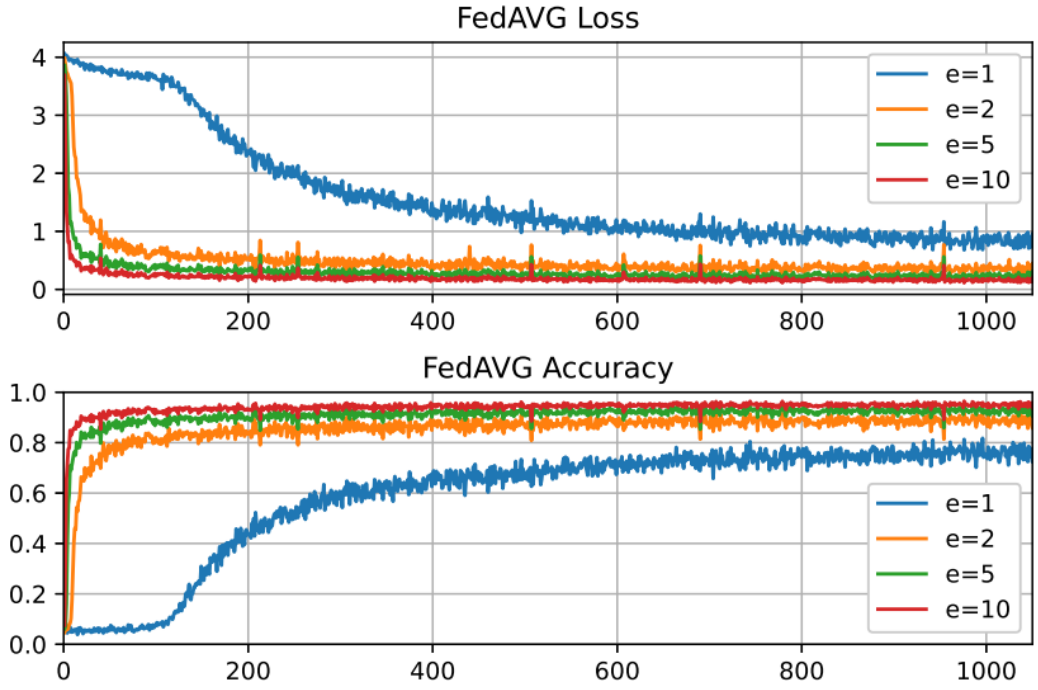


Figure 13: The impact of increasing local updated into Loss and Accuracy for Federated Average Algorithm

Considering only these findings, we can conclude that a high number of epochs yield better results. However, due to heterogeneity of data distribution, performing a high number of local updates could generate optimal local models. To investigate this line of thought, we evaluated the difference in collected metric values as well as the validation

Table 4: Communication Rounds necessary to achieve target loss and accuracy

0.75 Loss	**Epoch 1**	**Epoch 2**	**Epoch 5**	**Epoch 10**
FedAVG	744	43	14	7
FedADAM	614	39	11	5
75% Accuracy	**Epoch 1**	**Epoch 2**	**Epoch 5**	**Epoch 10**
FedAVG	305	189	111	57
FedADAM	258	163	87	48

loss and accuracy. For instance, in training with only one epoch, the mean accuracy of the last 200 rounds was 75.28% and validation 77.13%. With 10 epochs the mean accuracy was 94.95% and the validation accuracy was 83.96%. The difference between metrics using epochs equals ten increased 5.94 times the gap between accuracy and validation accuracy. In terms of loss, similar behavior was observed, the loss and validation loss for one and ten epochs were respectively: 0.8818 and 0.7565, and, 0.1613 and 0.6338. This comparison can be graphically analyzed in Figure 14.

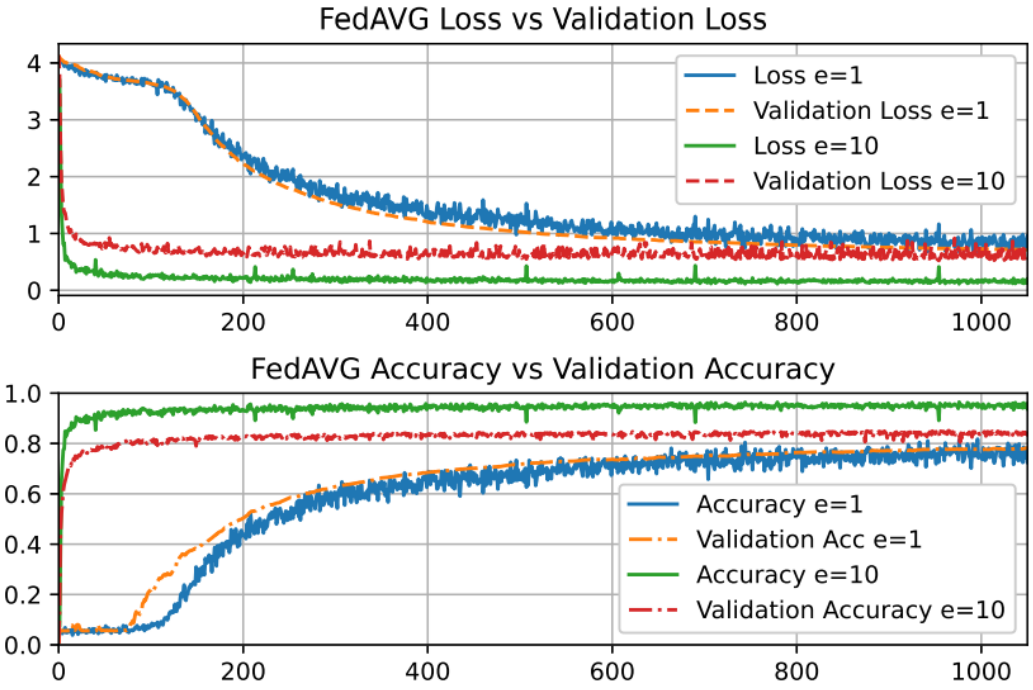


Figure 14: Comparative between loss/accuracy and validation loss/accuracy for epoch equals one and ten

We performed the same simulations using FedADAM and the increasing number of epochs also improved the results (Figure 15). The validation accuracy of ten epochs setting achieved the same result of one epoch. This demonstrates that increasing local updates for Adam optimizer can speed the convergence but no improved result was obtained at the end. Thus, the use of a high number of epochs for Adam optimizer is discouraged, since it will increase the time necessary to compute local updates.

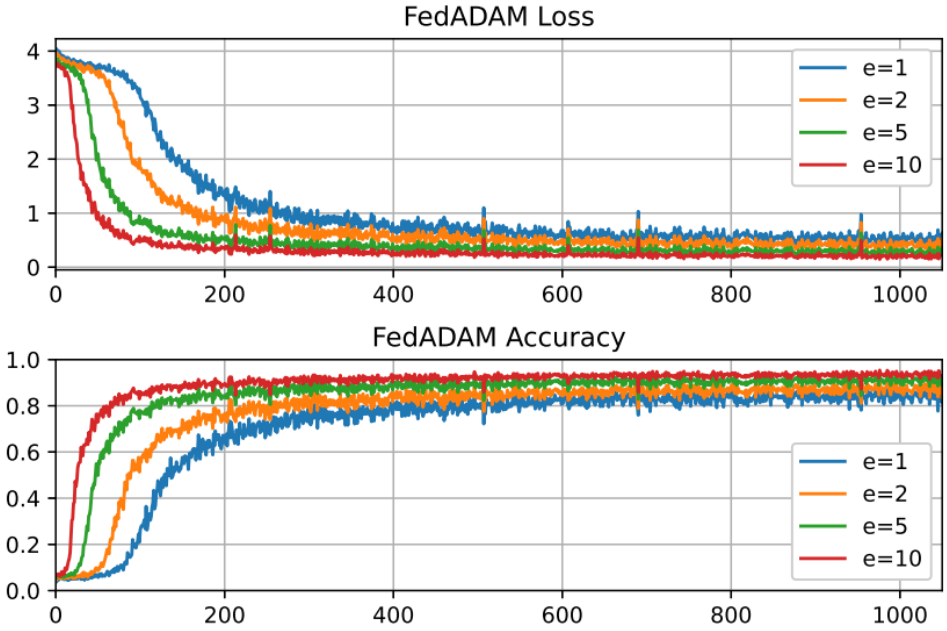


Figure 15: The impact of increasing local updated into Loss and Accuracy for Federated Adam Algorithm

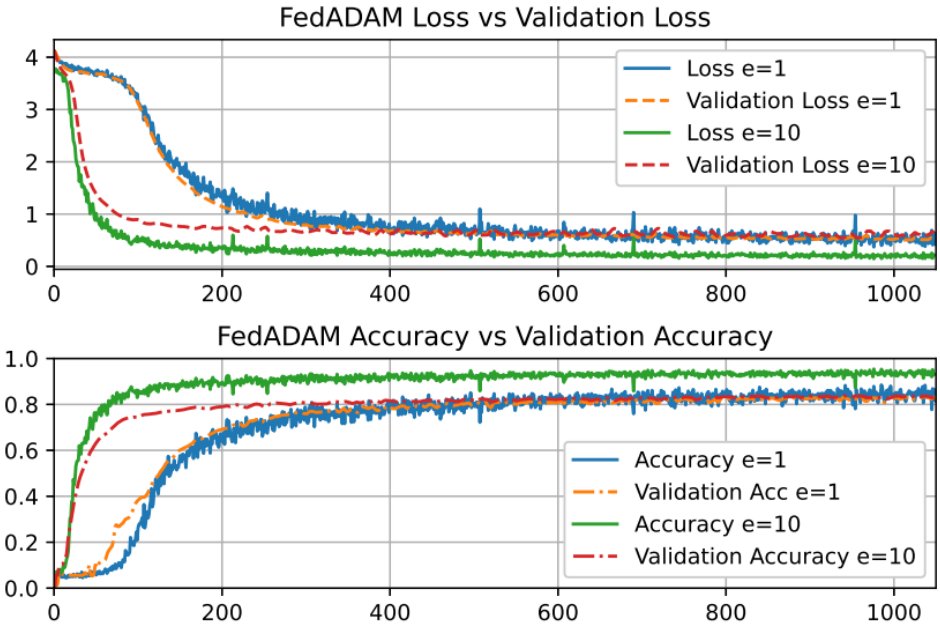


Figure 16: Comparative between accuracy and validation accuracy for epoch equals one and ten

We compared the Federated Average and Federated Adam algorithms as a function of the number of epochs. Results demonstrated that FedAVG produces better results for ten epochs, but when the number of communication rounds is not an issue, for instance in cross-silo settings, FedADAM provides faster convergence using one epoch per round. Considering the data collected through these simulations, we proposed an adaptation at the client-side, where the number of the local update should vary for less when using ten epochs settings. The goal is to diminish the overfitting and still increase the speed of

convergence. This experiment is described in the next section.

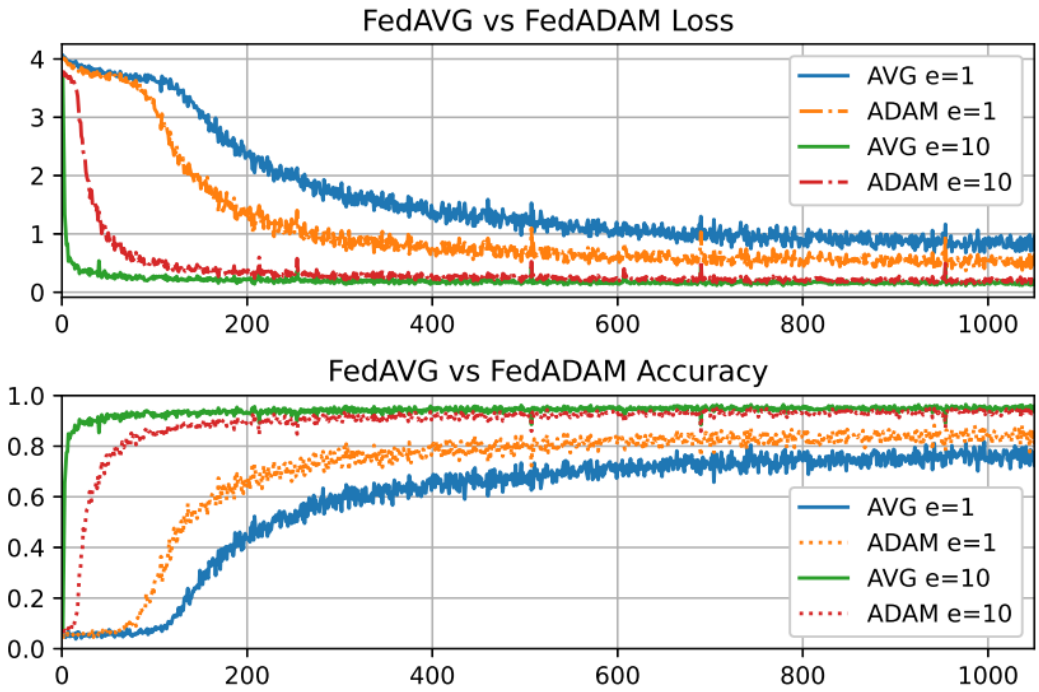


Figure 17: FedAVG works better than FedADAM for ten epochs

4.2 Proposed adaptation and results

In this section, we introduce a proposal for communication-efficient FL that tries to avoid communication overhead and overfitting due to the number of rounds pursued. The number of client epochs determines the amount of "sequential progress" (or learning) each client makes before updating the global model. A high number of epochs results in more local progress each round, this can manifest as a much faster per-round convergence rate. The risk of overfitting may be correlated to non-IID distribution of client datasets. The less similar to the "global" dataset each client dataset is, the more likely there will be "drift" (clients converge to different optimal points) [18] when using a high number of epochs during rounds closer to potential convergence.

In this scenario, [58] validation/test accuracy is less likely to improve. To reduce this negative effect, we added a controller at the client-side to vary the number of epochs and periodically reduces the local updates. We employed a Poisson distribution for that. We simulate two scenarios, the first with 1 epoch per round. The controller in this case periodically adds rounds using a Poisson distribution. In the second scenario, we considered 10 epochs per round, and the controller reduces the number of local updates, expecting to reduce the chance of yielding an optimum local. We performed 1500 rounds and set the probability of an event once in 1500 chances and for the other setting, we set ten times in 1500 rounds. Then, the limits of epochs were [1,10] for all settings.

Figure 18 shows the comparison of loss metrics for FedAVG before and after the adjustment for our first scenario, considering only one epoch as the default. As Poisson distribution will increase the number of local updates, but most of the time the probability will be concentrated around one epoch, the result improves just a little. It means that our approach doesn't boost enough the FedAVG. However, when considering the scenario with ten epochs that we analyze in the last section our method achieve considerable improvement. Figure 19 shows that using our variable number epochs, the gap between training and validation metrics is reduced. This behavior indicates that this model has less probability of overfitting. Even when it achieves less accuracy or higher loss, it still has better generalization and can be considered a better model according to these parameters.

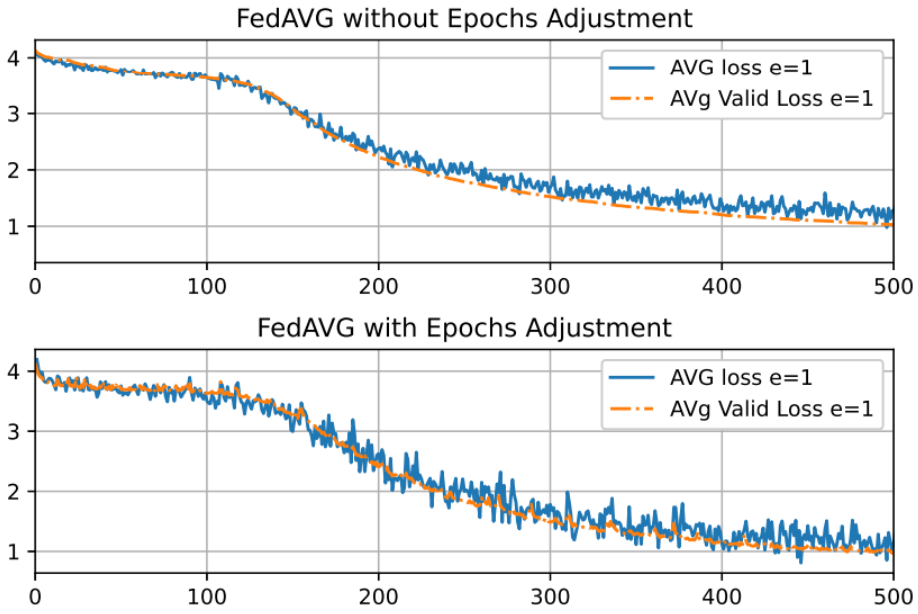


Figure 18: FedAVG Adjust using one epoch

The results of Figure 20 indicates that our approach can also be used with ADAM optimizer to improve model quality. Table 5 summarizes the results of both scenarios with FedAVG and FedADAM. For instance, the difference between loss and validation loss was 0.4541 before the adjustment and 0.0365 within. Also, the validation loss decreased 7.16% using the Poisson distribution controller. The same behavior happened with FedADAM, the difference between training and validation metrics reduces a lot and the comparison between validation accuracy with and without the adjustment showed a small decrease.

4.3 Chapter Conclusions

In this chapter, we first performed an empirical evaluation of the impact of local updates on FedAVG and FedADAM performance. We showed that increase in the number of epochs can speed up convergence but yields a high chance of producing overfitting (reducing the generalization). To mitigate the side effects of using too many local updates,

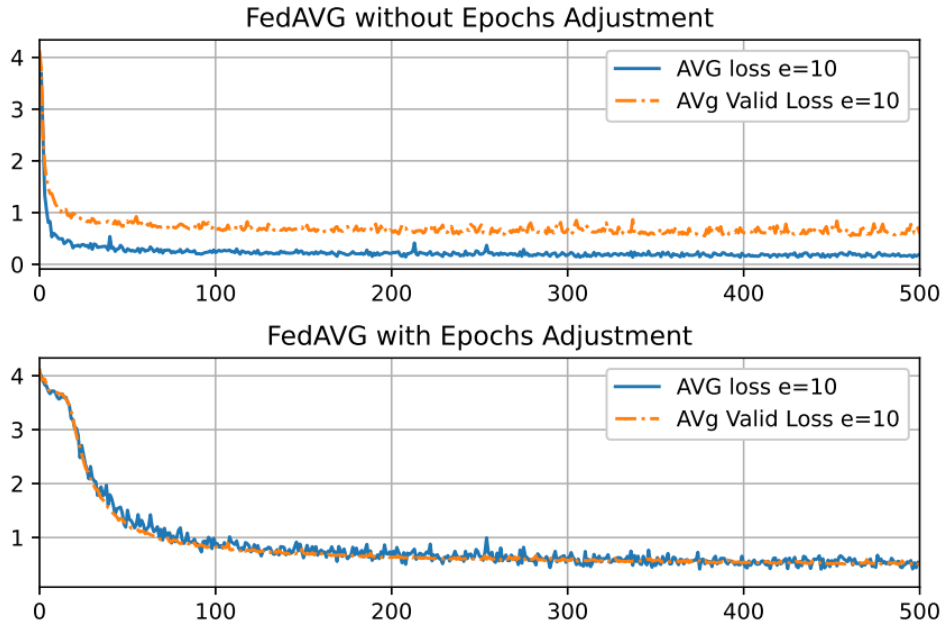


Figure 19: FedAVG Adjust using ten epoch

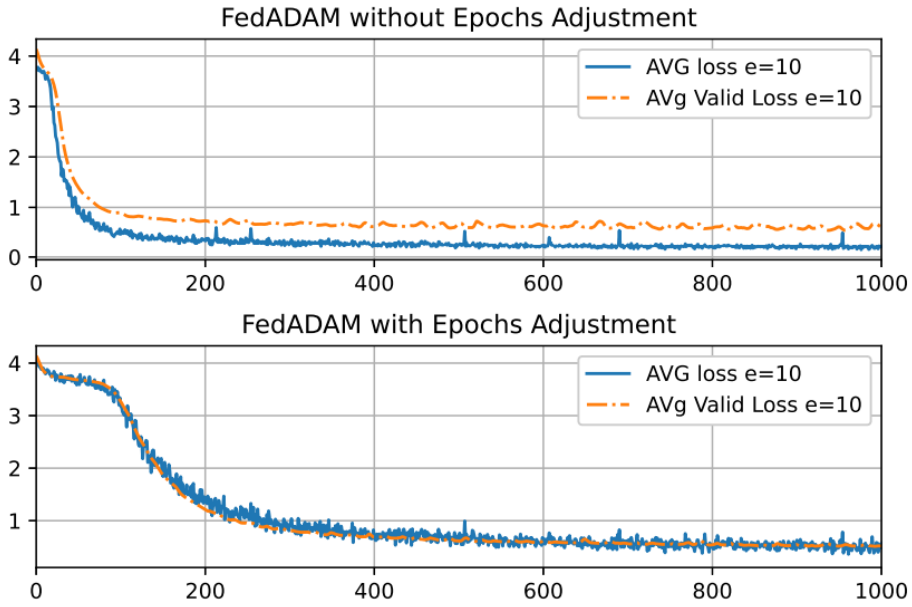


Figure 20: FedADAM Adjust using ten epoch

we further propose the utilization of a controller that increases or reduces the number of epochs periodically to improve the model generalization. The probability of the number of epochs used was configured with Poisson Distribution, we verified the advantages of the proposed methods reducing the difference between training and validation metrics. We conclude that the number of epochs shouldn't be defined statically; it depends on other constraints such as connectivity, mobility, computing resource. For example, if the client has weak connectivity or the network is overload, it should dynamically increase the number of local updates, but eventually reducing it to avoid model problems.

Table 5: Loss and Acc after our proposed adjustment

Metric between [200-300] rounds	**Loss**	**Valid Loss**	**Acc**	**Valid Acc**
FedAVG 10 Epochs	0.2046	0.6587	93.74%	82.61%
FedAVG 1 Epoch	1.9806	1.8252	52.61%	57.66%
FedAVG 10 E Adjust	**0.6480**	**0.6115**	**80.78%**	**80.14%**
FedADAM 10 Epochs	0.3175	0.6963	90.32%	79.86%
FedADAM 1 Epoch	1.0906	0.9330	71.11%	73.33%
FedADAM 10 E Adjust	**1.0962**	**0.9703**	**70.87%**	**72.43%**

Next chapter, we extend the evaluation by adding more variables to the simulation as more sophisticated client selection criteria using Cluster-based client selection mechanism for FL applications that considers Model and data size information. We also detail a performance evaluation where we demonstrate that our approach can maintain the fast convergence and also contribute to reducing the number of rounds between clients and the aggregation server.

CHAPTER 5

MESFLA

FL emerges as a promising solution to enable collaborative model training while preserving the privacy and autonomy of participating clients. FL facilitates collaborative model training by enabling data to be trained locally on devices, eliminating the need for individual information sharing among clients. A client selection mechanism strategically chooses a subset of participating clients to contribute to the model training in each learning round. However, an efficient selection of clients to participate in the training process directly impacts model convergence/accuracy and the overall communication load on the network.

In addition, FL faces challenges when dealing with non-Independent and Non-Identically Distributed (non-IID) data, where the diversity in data distribution often leads to reduced classification accuracy. Hence, designing an efficient client selection mechanism in a scenario with non-IID data is essential, but it is still an open issue. This chapter proposes a Model Efficiency through Selective Federated Learning Algorithm called MESFLA. The mechanism employs a Centered Kernel Alignment (CKA) algorithm to search for similar models based on data weight or similarity between models, *i.e.*, grouping participants with comparable data distributions or learning objectives. Afterward, MESFLA selects the clients with more relevance in each group based on data weight and entropy. Our comprehensive evaluation demonstrates MESFLA's superior performance over traditional FL algorithms. Our results show an accuracy improvement and a minor loss in each client aggregation of the new global model sent to clients with a difference of 3 rounds using the Data Weight in comparison with the other selection methods.

5.1 Introduction

FL emerges as a crucial solution to provide a privacy-preserving property with reduced communication cost for future ML applications [59]. FL provides a decentralized ML paradigm, which leaves the training data distributed on mobile devices and learns a shared model by aggregating locally computed updates [60]. In FL operation, the client selection mechanism deployed at the aggregation server selects a subset of participating devices, also called clients, to contribute to the model training in each communication round [61]. These clients receive the global model, conduct training based on their local data, and then share their model parameters instead of transmitting their raw sensing data, keeping the raw data on the client devices [62]. On an aggregation server, only locally computed updates and analysis results are received, and aggregated for an improved global model that benefits from distributed learning, such as achieving improved accuracy or better generalization [63]. Hence, FL allows continuous learning by adapting the ML model without sharing raw data [64].

The mechanism must choose a subset of clients eligible to participate in the upcoming learning rounds, where the mechanism must remove clients who do not add value to the training to improve the global model while reducing the waste of computation resources [65]. In this sense, selecting clients with valuable data samples is crucial, as it both saves computational resources and excludes data that no longer benefits the global model's improvement. [61]. Random client selection is commonly used in traditional FL algorithms, but client devices often have heterogeneous data distribution, and randomly selecting them might lead to some bias, low-performance metrics or some challenges to achieve ML model's convergence. It is also important to find techniques that reduce the number of communication rounds and the size of the transmitted messages [4]. The client selection mechanism is a critical component of the FL training process, as it ensures a diverse and representative data sample from various clients. Hence, client selection is not an easy task, and there are many issues to consider for defining the best client selection.

Furthermore, the client selection mechanism faces challenges when dealing with non-independent and non-identically distributed (non-IID) data scenarios, as datasets may have statistical heterogeneity. This diversity in data distribution often leads to reduced classification accuracy. One approach to mitigating the impact of non-IID data on model gradients involves assessing the similarity between trained model representations. In this way, clustering algorithms are an essential tool to group participants with comparable data distributions or learning objectives, creating clusters with similar models. Hence, it is possible to select clients with more relevance in each group, *i.e.*, clients that contribute to the model training in each learning round, improving the accuracy and convergence.

In this thesis, we propose a cluster-based client selection mechanism for FL applications that considers model and data size information, called MESFLA. The mechanism considers the clusterization and client selection steps. At the clusterization step, the mechanism considers a clusterization algorithm to group clients based on the weight data

of each client. Afterward, at the client selection step, the mechanism selects the clients with more relevance in each group. In addition, the aggregation server evaluates the rate of change in accuracy to decide when to return the new aggregated Model to all clients, saving network resources while maintaining the global Model during many rounds. Evaluation results demonstrate that MESFLA achieves model efficiency by using significantly fewer communication rounds between clients and aggregation servers and exhibiting a faster recovery system than other approaches. The main research contributions of this chapter can be summarized as follows:

1. Cluster-based client selection mechanism for FL applications that considers Model and data size information.

2. A detailed performance evaluation where we demonstrate that our approach can maintain the fast convergence and also contribute to reducing the number of rounds between clients and the aggregation server.

5.2 Model Efficiency through Selective Federated Learning Algorithm (MESFLA)

This section introduces a cluster-based client selection mechanism for FL applications that considers model and data size information called MESFLA. The proposed mechanism consists of two steps, namely, clusterization and client selection. At the clusterization step, the mechanism considers a clusterization algorithm to group clients based on the weight data of each client. Afterwards, at the client selection step, the mechanism selects the clients with more relevance in the each group. In the following, we introduce the system model and MESFLA operations.

5.2.1 Scenario Overview for Clustering and MESFLA operations

We considered the traditional FL architecture, where at each communication round, a set of Client k is selected to receive the global model, perform the training based on its Dataset *Dn*. The clients improve the Model *Mn* by training with its specific data. Afterward, the model updates, *i.e.*, learned parameters or gradients, are sent periodically to the central server [66]. The aggregation server applies a given aggregation policy, such as FedAVG. For instance, FedAVG computes an average of the shared local models at edge servers to produce an accurate global model. Finally, the updated global model is distributed to the selected clients [67].

The client selection mechanism must select a subset of clients to participate in the upcoming learning rounds. The mechanism must select a given client k with valuable samples to reduce the waste of computation resources, and it must remove the clients

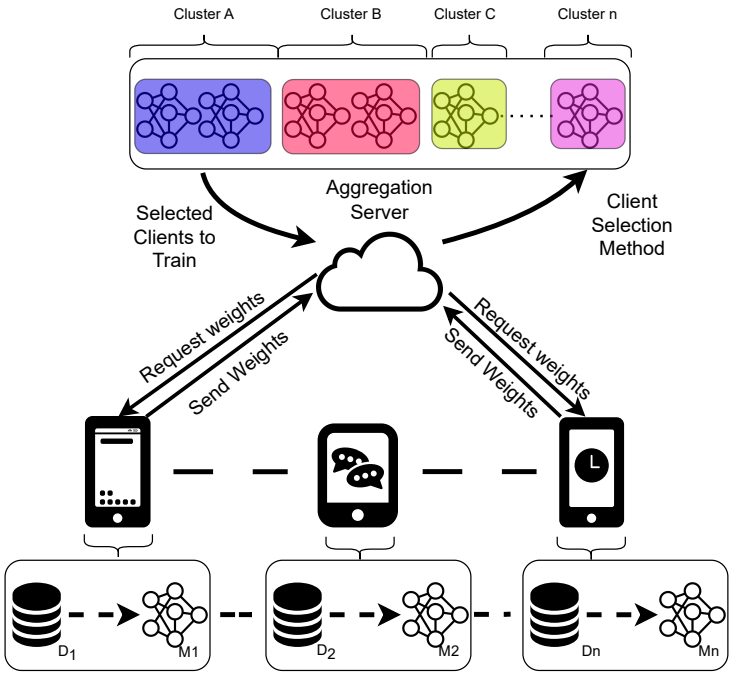


Figure 21: MESFLA operational overview

whose data are no longer critical for the model training. In addition, FL is subject to data that are not independent or have different statistical distributions, *i.e.*, non-IID data scenarios. This statistical heterogeneity of Dataset Dn results in lower classification accuracy, where non-IID data over model contributions can be addressed by measuring the similarity between trained model representations. In this context, clustering algorithms play a vital role in FL by grouping participants with similar data distributions or learning objectives, enabling to group similar models.

Figure 21 shows the MESFLA operation, where each device has its own collected data (D), *i.e.*, the usual preferences and data pattern recorded tipically in that device due to personal usage patterns. Each device uses its own collected data D to train a local model M associated with that device. Then each device sends the trained model together with the dataset size to the central server. In this way, MESFLA collects all local model M from all the available devices, and the central server aggregates each model based on a given aggregation policy, such as FedAVG, providing a global model. In addition, the central server clusters that received model M were into similar groups/clusters. Afterward, MESFLA selects the clients with more relevance in each group, which is the client with the large dataset. Table 6 summarizes the list of main symbols used to introduce the MESFLA mechanism.

5.2.2 MESFLA Operation

MESFLA Operation is divided into **Clusterization** and **Client Selection** steps. In the Clusterization step, we consider Centered Kernel Alignment (CKA) implementation on the models to determine which group each user belongs to, *i.e.*, the set of clients

Table 6: List of Symnols

Acronyms	**Description**
Dn	Dataset
Gm	Global Model
Mn	Model
Wfl	FL model
Wk	Client weight within the cluster
Wi	Cluster weight
Wj	Client weight
C	Set of clients
k	Client
P(k)	Probability of k
Sn	The sum result in the total data from all samples.
Sk	Number of data from that sample
t	Threshold
w	Set of w rounds

k converges into groups based on the model similarity after their training [68]. In its operation, a given client k sends its client weight Wk to the server, which clusters C these parameters with the aim of grouping the set of k clients based on model similarity. For instance, clients have a correlation based on the values of model gradients, which is used to search for the most similar internal representations between models (*i.e.*, clients) [50]. In this way, CKA provides insights into how well models capture similar patterns of client weight Wk and structures in the data, even when trained independently or on different tasks. Algorithm 1 illustrates the Clusterization step performed by the MESFLA mechanism operations.

The CKA uses different methods to measure the similarity between the matrices. In this way, we use the Dot Product-based Similarity, which relies on the Eq. 5.1 of the relative dots products of the samples where H is the centering matrix. The Eq. 5.1 presents the empirical estimator for the Hilbert-Schmidt Independence Criterion (HSIC), a non-parametric measure of the dependency between two sets of variables, *i.e.*, X and Y, represented within kernel matrices K and L. The elements $K_{ij} = k(X_i, X_j)$ and $L_{ij} = l(Y_i, Y_j)$ are derived from kernel functions k and l. That means mapping pairs of data points into a high-dimensional feature space, capturing their similarities. The matrix H is a centering matrix that adjusts the kernels to have a zero mean. The trace operation, $\text{tr}(KHLH)$, in the numerator, captures the alignment of the transformed data points. Divided by the square of the number of observations minus one, $(n-1)^2$, this formula normalizes the value to account for sample size, providing a scaled estimate of dependence robust to the number of data points in the sample.

Quantifying the similarity between features involves calculating the sum of squared dot products across all pairs. Otherwise, CKA involves calculating the alignment between the centered representations of two models by comparing their kernel matrices. The kernel matrix represents the pairwise similarity between data points in the high-dimensional space.

$$HSIC(K, L) = \frac{1}{(n-1)^2} tr(KHLH), \tag{5.1}$$

The alignment is then normalized to provide a similarity score between 0 and 1 based on Eq. 5.2, where 1 indicates perfect alignment. The Eq. 5.2 uses the HSIC to compare the similarity of K and L matrices. That matrix typically represent features in ML models, by normalizing HSIC's measure of dependency between the two sets of variables. The CKA offers a scale-invariant index, ensuring that the similarity assessment is not affected by the magnitude of the data. This normalization, defined as $\sqrt{HSIC(K, K)HSIC(L, L)}$, allows for a direct and equitable comparison of feature representations across different models or data transformations that provices for CKA a valuable tool in the analysis and interpretation of complex machine learning systems.

$$CKA(K, L) = \frac{HSIC(K, L)}{\sqrt{HSIC(K, K)HSIC(L, L)}} \tag{5.2}$$

Algorithm 1: Clusterization step

```
foreach client do
    Get user weights for the client;
    Wfl ← Wc.append, Client.name;
end
// Cluster all clients' weights
Call clusterUsers();
return Clustered clients weights;
Function clusterUsers():
    // Perform operations on user weights for a specific client
       if needed
    foreach Wi in Wfl do
        listWi ← +∑_{k=n}^{W} Wi;
        foreach Wk in Wfl do
            listWj ← +∑_{k=n}^{W} Wj;
        end
        if Wi ≥ listWi - listWk and Wi ≤ listWi + listWk then
            Clusterlist ← client.name;
        end
    end
```

In the client selection step, the edge server considers the data size as the metric

to select a given client k that is more relevant to each group. For this purpose, the server receives the model *Mn* for each client and assigns a client weight within the C weight *Wi*. Afterward, we use the probability of each client entering the training based on its greater relevance than others in the same C cluster. In this way, we can use the probability *P(k)* computed based on Eq. 5.3 we ensure that the best candidates are more likely to participate in the training to prevent overfitting. The *Sk* means the total number of times this k has priority, and *Sn* denotes the sum of times each k appears on this array sum to the *Sk*. This method helps to balance the influence of each client, ensuring that even those with smaller datasets have a fair chance of being selected. This mitigates the risk of overfitting the model to the characteristics of larger datasets alone by balancing the weights with this probability *P(k)*.

$$P(k) = \frac{S_k}{\sum_{n=0}^{N} S_n} \tag{5.3}$$

Once the model is trained, it is evaluated on a test dataset to determine its performance, where model aggregation increases the accuracy results on each round evaluation because the round behaves like epochs. MESFLA considers a window (*i.e.*, a set of w rounds) to evaluate the variation of the accuracy value, which can be compared with the performance of the model trained in the previous window [49]. Specifically, after w rounds, the aggregation server evaluates if the model accuracy changed less than a given threshold t. If so, the server sends a new model for the clients, reducing the amount of data required to transfer for model transmission. In this way, the MESFLA relies on a convergence point as a way to reduce the bandwidth consumption to transfer a new model at each round. It is important to mention that MESFLA requires a threshold t and a window w, where these values may need to be adjusted based on the specific characteristics of the dataset and the network environment.

5.3 Evaluation

This section describes the scenario, including the framework, database, and simulation details. We also discuss the obtained results in terms of computational effort, loss, and the global model's accuracy.

5.3.1 Simulation Description

We conducted simulations using the Personalized Federated Learning Platform, a framework available on GitHub of TsingZ0[1]. We chose this framework due to its flexibility and the various aggregation models that were implemented. In this way, we use such a

[1]TsingZ0/PFL-Non-IID: Personalized federated learning simulation platform (https://github.com/TsingZ0/PFL-Non-IID)

platform to support our scenario running a server with the following specifications: 13th Gen Intel i9-13900K (32) @ 5.500GHz, eight cores, 128GB RAM. We considered the MNIST dataset for classification in FL, which is a widely used dataset to train and test for model validation, particularly for tasks related to image classification. Specifically, the MNIST dataset is a widely used collection of 28x28 pixel grayscale images of handwritten digits (0 through 9). MNIST Comprises a training set of 60,000 images and a test set of 10,000 images, which serves as a benchmark for testing and validating ML algorithms. We consider a ML model with two convolutional layers with filter sizes of 5x5. A 2x2 max-pooling operation succeeds each convolutional layer. Table 7 summarizes the main evaluation parameters.

We consider that data exhibit non-IID characteristics due to data heterogeneity across various FL applications. In this sense, numerous distributed ML training approaches encounter substantial degradation of accuracy due to the data quantity and category distribution disparities within non-IID data. In the non-IID context, we use label distribution skew to characterize the local data distribution among clients, resulting in varying label proportions [69]. Specifically, we sample a proportion $p_{k,i} \sim Dir(\beta)$ that represents the instances of class k to the client, where $Dir(\beta)$ is a Dirichlet distribution with a concentration parameter $\beta = 0.2$ [70].

In our evaluation, we analyze the impact of different aggregation policies by comparing their behavior of using a cluster approach with different client selection methods. Specifically, we consider FedAVG and Model-Contrastive Federated Learning (MOON) [70] as the aggregation policies. The FedAVG uses the client's averages to aggregate their parameters to the newest global model in the FL system, providing fair processing of the weights for all clients who do the training step. For this purpose, we compare the behavior of this strategy with our clustering method. On the other hand, MOON uses the local dataset, such as the weights of each client relevant in the aggregation step, providing efficiency when using the parameter in the update step. Then, they send extra clients' weights relevant to the system data. In other words, each client has different relevant data when their parameters are aggregated in the newest global model Gm. Therefore, the primary approach of the MOON is to declare a disparity of each client's contribution to the global Gm model for better use.

In addition, we evaluate the cluster approach based on CKA with different client selection methods, namely MESFLA Default, MESFLA Data Weight, and MESFLA Entropy Weight. Specifically, MESFLA Default means a random client selection for each group, which serves as a baseline method that simply selects a random subset of clients to participate in each round of training. On the other hand, MESFLA Entropy selects the set of clients based on the number of labels as input to compute the entropy for each user, such as introduced by Sousa et al.[49]. Finally, MESFLA Data Weight means the client selection introduced in Section 3.2.

We compare these algorithms with commonly used metrics for FL classification, namely, processing level of the device, accuracy, and loss. The **Accuracy** metric is obtained by the number of hits (positive) divided by the total number of examples, which

Table 7: Simulation Parameters

Characteristics	**Values**
Cluster Method	CKA
Total Number of Clients	100
Selection methods	Random Selection, Entropy, and Data size
Number of selection clients	20% clients each round
Server strategies	FedAVG
Dataset	MNIST
Rounds	100 rounds
Local Training	1 epoch
Number of labels	10 labels/classes

is used on data with the examples for each class and when they miss. In the case of disproportionate classes, it gives a false impression of good performance, delivering a flawed result. The **Loss** metric compares the target and predicted output values, which helps see how the neural network models the training data. The computational effort metric is computed based on the average time of each round and the maximum TeraFlops that an RTX 4090 has.

5.3.2 Simulation Results

We start our evaluation by analyzing the behavior of different aggregation strategies, where the ideal result is searching for a clustering method that can work with high accuracy and low loss values, allowing us to understand how good it is in our estimated recovery time. Figure 22 demonstrates each computer processing time in terms of Teraflops for different evaluated categories. The main goal is to achieve the model efficiency performance of the clients by considering time and calculations. By analyzing the results, we can observe that FedAVG costs less in computation processing than MOON, regardless of the client selection approach. For instance, MOON shows more time to process and analyze the new model of FL in each round, which has higher processing. This result shows the best strategy for our approach, leading to a strategy that uses the power of clients' datasets as a relevant argument to evaluate them with more processing recurses. We can also do it on devices with low processing when we know these values are necessary to train them in a high-level cloud. In this way, we need to see how the strategies are forming in the accuracy of these strategies when choosing them.

It is essential to highlight that the aggregation server evaluates if the model accuracy changed less than a given threshold t. If so the server sends a new model for the clients, which might impact the amount of data sent for model transmission. Figure 23 shows the accuracy evaluation of different client selection approaches with FedAVG as an aggregation policy in the simulation with a threshold t of 1% convergence in the last

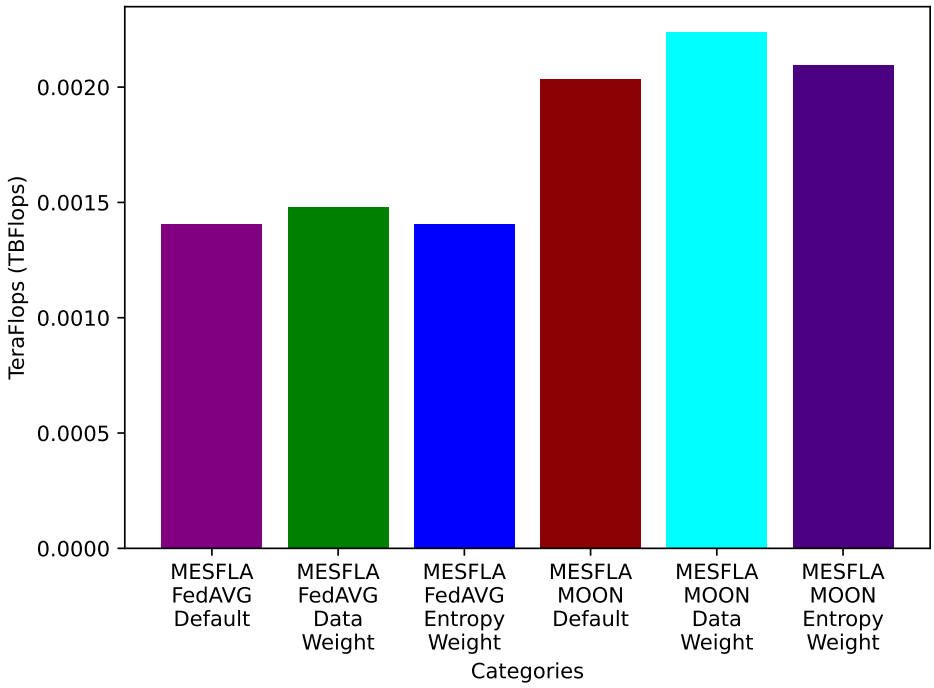


Figure 22: Processing results

five rounds. This result shows how MESFLA Data Weight improves the results with a convergence variety.

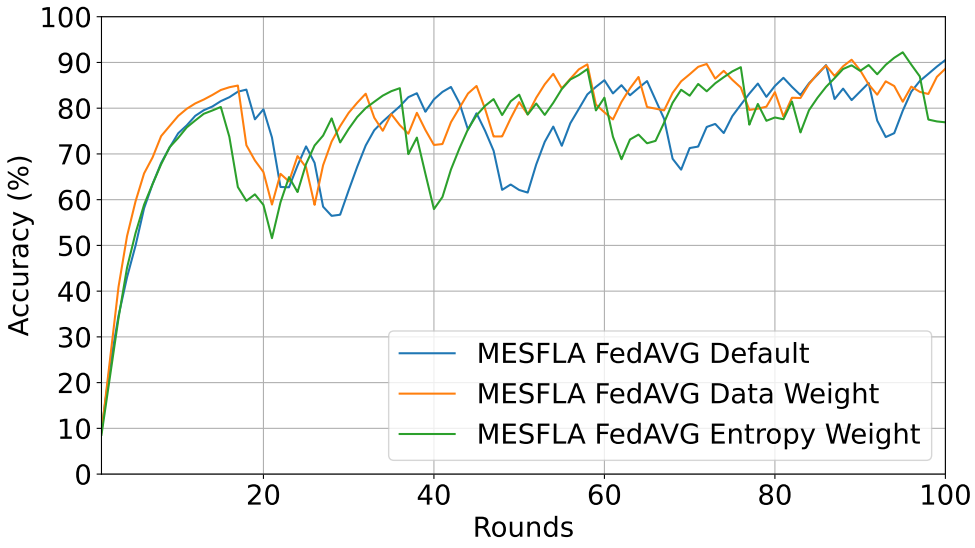


Figure 23: Accuracy results with FedAVG as aggregation policy with 1% of convergence

Figure 24 shows the evaluation of different client selection approaches with FedAVG as an aggregation policy and a threshold t of 5% in the last five rounds. It is noticed that accuracy falls each time that the global model Gm is sent to devices for each one of our categories. Our approach evaluates the accuracy of every round and as soon as the convergence reaches five rounds without more than 5% of progression, it sends the

global model Gm to the clients. By comparing the results of Figures 23 with 24, we could see that threshold t of 1% proves to have a longer time to convergence than using 5%, as well as the model updates are sent more frequently, as evidenced by the more oscillatory nature of the accuracy lines across rounds. This leads to a higher network cost due to increased transmissions required to achieve this stringent convergence level. On the other hand, Figure 4 considers a threshold t of 5%, leading to a smoother trajectory for accuracy improvements. Hence, this indicates that the model under a 5% threshold t requires fewer updates to maintain or improve accuracy, thereby reducing network expenditures.

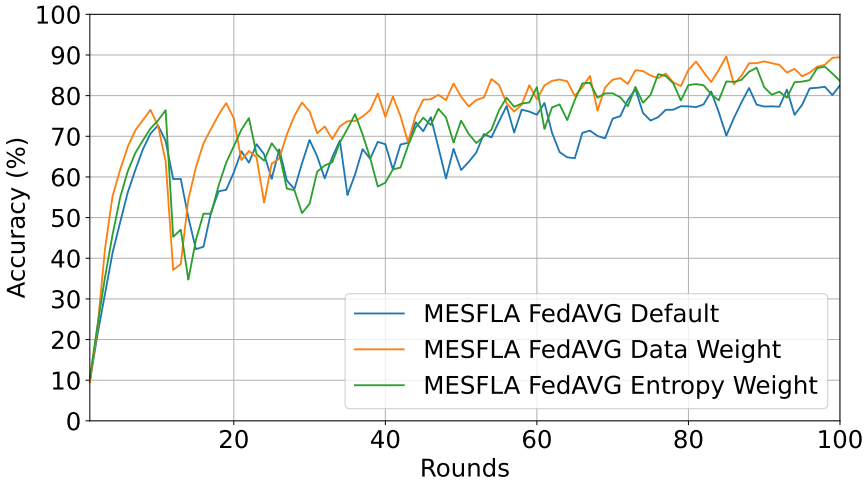


Figure 24: Accuracy results with FedAVG as aggregation policy with 5% of convergence

Figure 25 shows a Loss results with different client selection approaches with FedAVG as aggregation policy for a scenario with convergence threshold t of 5%. There is a round at the beginning of training with a loss value above 0.75 for data weight, followed by other approaches. However, the evaluation goes up again in sequence because the clients receive the global models Gm. In the following rounds, the impact of local model re-training is lower. Figure 25 demonstrates the selection with Data Weight having the best evaluation with less round of training to reach an efficient global model Gm. In addition, the entropy method was close to this value during a few more rounds in means of 3 or 4 rounds more to reach the same evaluation, an improvement of 20% of our FedAVG Default. That could easily rearrange itself after increasing values. In this way, we maintain the random selection (Default) in a random approach to calculate new parameters for the scenario, meaning our baseline to validate the selection improvement.

Figure 26 shows accuracy results with different client selection approaches with MOON as aggregation policy for a scenario with convergence threshold t of 5%. By analyzing the results, we can observe that the first threshold occurs around 78% in means in each code run, which means an almost 5% improvement compared to FedAVG approach. However, the selection shows another behavior of using this strategy with our approach of selection by Datat weights the data of most of the cases the global model Gm does not reach the threshold t in the middle rounds, becoming a more stable system with around 80 to 90 accuracy during rounds 50 and 80. Moreover, in another way, this strategy also

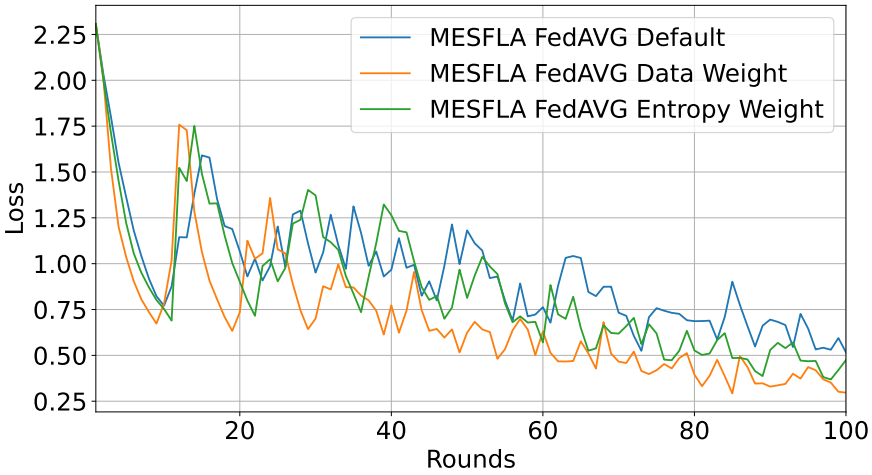


Figure 25: Loss results with FedAVG as aggregation policy with 1% of convergence

sends the global model Gm to the clients a few times.

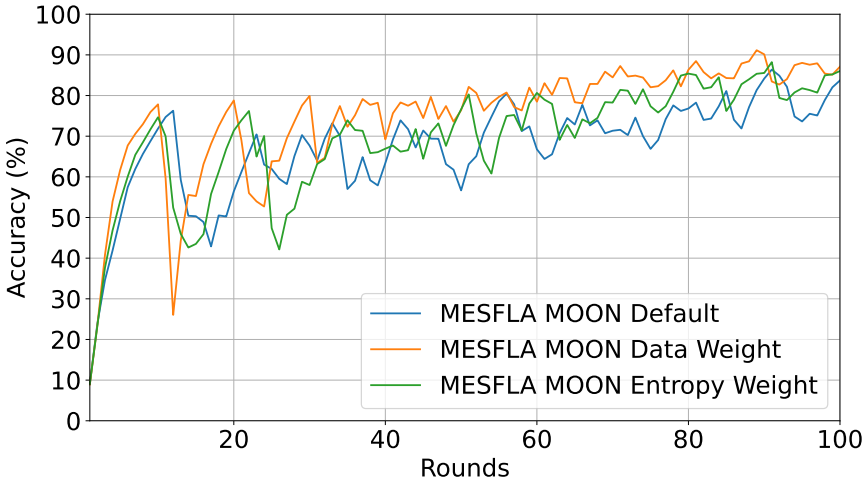


Figure 26: Accuracy results with MOON as aggregation policy with 5% of convergence

Figure 27 shows loss results with different client selection approaches with MOON as aggregation policy for a scenario with convergence threshold t of 5%. By analyzing the results, we can observe that the results of the default method are similar to those of others in the initial rounds. However, the MOON Data Weight gets more useful results after some rounds, having a peak of almost 2 of Loss after the first time they reach the threshold t. However, data weight and entropy evaluation demonstrate less time or a slight loss and, in a few rounds, a total recovery to improve these metrics, reaching the exact evaluation of the last threshold t in only around seven rounds. Further, the result shows an improvement to MESFLA using this strategy and an excellent approach when treating discalceate data in clients.

Table 8 summarizes the number of rounds required to converge the model with different client selection approaches and FedAVG and MOON as aggregation policy. By

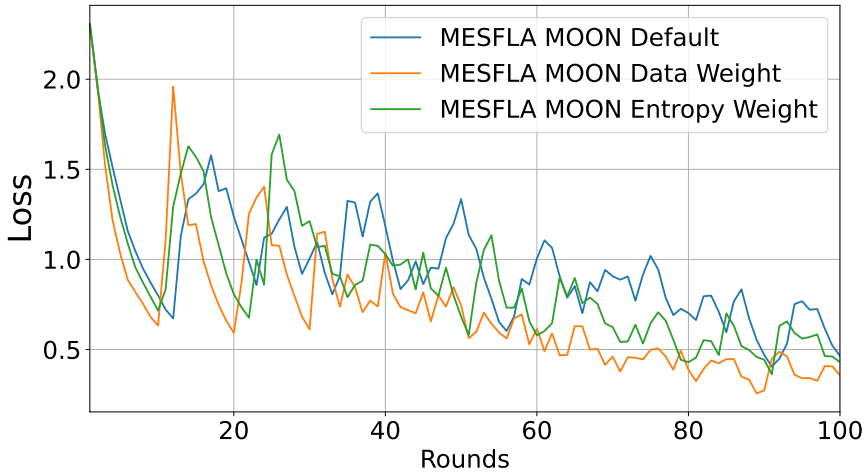


Figure 27: Loss results with MOON as aggregation policy with 5% of convergence

analyzing the results, we can see that the FedAVG requires more convergence rounds than MOON training. This is because the MOON update weight, which is not necessarily needed in all upgrades, transforms into a problem to reach our convergence of 5 rounds of improving by 5 %. It is important to mention that there is a variance in the results, as we can see in Figure 26, which compares the rounds around 40 and 60. We observe an occasional worsening in the evaluation, primarily when novel data for the global model Gm is not used. This complication arises when the best fit for training is not employed.

After these results, we must compare the primary purpose of MESFLA and the maintenance of the results using fewer sendings of the global model Gm to the client side. We search for a "good" evaluation as we have a low convergence time. . However, we can see that the FedAVG has more convergence steps during the MOON training. That was caused by his update weight, which is not necessarily needed in all upgrades, transforming into a problem to reach our convergence of 5 rounds of improving by 5 %.

Table 8: Simulation Results of Convergence sending.

	Category	Rounds to Convergence
FedAVG	MESFLA FedAVG	12.6
	MESFLA FedAVG Data Weight	14.6
	MESFLA FedAVG Entropy Weight	12.6
MOON	MESFLA MOON	8.3
	MESFLA MOON Data Weight	11.0
	MESFLA MOON Entropy Weight	9.4

5.4 Discussion

The threshold t of 1% as a criterion within the last five rounds can enhance model precision, as shown in Figures 24 and 23. However, this approach requires additional communication rounds, potentially leading to increased communication costs and processing consumption. In contrast, the threshold t of 5% achieves quicker model stability, but possibly at the cost of model accuracy. These findings have significant implications for FL deployment, where devices with varying capacities and network conditions can be highly dynamic.

Setting the convergence threshold to 1% typically causes the model to interact and refine its parameters with each round until it achieves a precise level of accuracy. This rigorous standard can enhance model precision by allowing the model to learn and adjust from a greater volume of aggregated updates. However, this meticulous process requires frequent client and server communication to exchange model updates. The Communication cost consideration is crucial in areas where network traffic may be congested, potentially leading to delays and increased operational costs.

The model achieves an acceptable accuracy level with fewer communication rounds with a threshold t of 5%, accelerating model stabilization. This is because the model requires fewer communication rounds to reach an acceptable level of accuracy, allowing for faster deployment of the updated global model. This experience is advantageous in dynamic settings where faster response times might be necessary. It may come at the expense of the model's overall accuracy. Such a trade-off might be acceptable in applications where speed is critical and absolute precision is less vital. In summary, a lower convergence threshold could strain the network and the devices, leading to processing use and potential network bottlenecks. Conversely, a higher threshold could save resources but compromise the quality of the service.

Furthermore, the loss patterns observed in Figures 25 and 27 highlight client selection methods' significant impact on both convergence speed and model generalization across diverse data distributions. Data Weight and Entropy-based selections achieve a reduction in training rounds and loss values, accelerating the stabilization of the model. The reduction in loss under the Data Weight strategy suggests that this method is particularly effective in scenarios where data distributions are heterogeneous. By assigning weights to clients based on the relevance of their data, the aggregation process becomes more representative of the overall data landscape, enhancing the model's generalization capabilities. Limiting the convergence threshold to 5% prevents the models from overfitting, as they do not continue to learn once an acceptable loss is reached.

The FedAVG approach demonstrates a more consistent decline in loss values when Data Weight is applied, indicating a smoother and potentially more stable path to convergence. This behavior has benefits scenarios with heterogeneously distributed data, where the learning process favors a gradual and steady approach. The more uniform decline in loss values across rounds suggests that FedAVG, combined with Data Weight

optimization, may be more suitable for environments where predictable and reliable model performance is critical. Therefore, the choice between the two strategies should be based on the application's specific needs. That is when the priority is stability and reliability with FedAVG or faster adaptability with MOON.

The convergence data offers direct conclusions about the communication cost in FL algorithms. For FedAVG, applying Data Weight increases the communication rounds needed, potentially leading to higher network usage and potentially more significant costs. The Entropy Weight method, however, does not impact the frequency of communication compared to the baseline, suggesting it might be a more efficient choice within the FedAVG framework when communication overhead is a concern. In contrast, MOON's mean convergence figures are consistently lower, reflecting its communication efficiency. The MOON is a preferable algorithm for scenarios with limited bandwidth or where minimizing communication is crucial. The results invite a focused discussion on choosing FL strategies that align with specific network efficiency needs. The MOON highlights its Entropy Weight variant, potentially more suitable for communication-constrained environments.

5.5 Chapter Conclusion

This chapter introduced MESFLA, a novel cluster-based client selection mechanism for federated learning applications that effectively addresses the challenges of non-IID data distributions while optimizing communication efficiency. By leveraging the Centered Kernel Alignment (CKA) algorithm, MESFLA successfully clusters clients with similar data distributions and selects the most relevant clients from each cluster based on data weight and entropy metrics.

Our comprehensive evaluation demonstrated that MESFLA significantly improves model performance compared to traditional federated learning approaches. The incorporation of data weight as a selection criterion proved particularly effective, showing accuracy improvements and reduced loss with fewer communication rounds between clients and the aggregation server. The experimental results also revealed valuable insights regarding convergence thresholds, where a 5% threshold offered a better balance between model accuracy and communication efficiency than stricter criteria. The comparative analysis between FedAVG and MOON aggregation policies showed that MOON generally requires fewer convergence rounds, making it more suitable for bandwidth-constrained environments. However, FedAVG combined with Data Weight selection demonstrated more stable convergence patterns, which may be preferable in scenarios where predictable model performance is crucial.

MESFLA's monitoring system at the aggregation server effectively evaluated the rate of change in accuracy, enabling intelligent decisions about when to distribute updated global models to clients. This approach resulted in significant communication savings while maintaining competitive model performance, addressing one of the critical chal-

lenges in federated learning deployments. These findings contribute to the advancement of federated learning by offering a flexible and efficient client selection mechanism that balances model performance with system resource constraints. MESFLA provides an effective solution for practical federated learning implementations in diverse network environments, particularly those dealing with heterogeneous data distributions across client devices.

CHAPTER 6

Conclusion

This thesis presents two significant contributions to the field of FL. First, we conducted a comprehensive empirical evaluation of local updates' impact on FedAVG and FedADAM performance. Our analysis revealed that while increasing the number of epochs can accelerate convergence, it significantly increases the risk of overfitting, thereby reducing model generalization capabilities. To address this challenge, we developed an adaptive controller that dynamically adjusts the number of epochs using a Poisson Distribution. This approach successfully improved model generalization by reducing the gap between training and validation metrics.

Our findings demonstrate that static epoch configuration is suboptimal in FL environments. The optimal number of local updates should instead adapt to various constraints including connectivity quality, client mobility patterns, and available computing resources. For instance, in scenarios with weak connectivity or network congestion, the system should dynamically increase local updates while implementing periodic reductions to prevent model degradation.

The second major contribution introduces MESFLA (Model Efficiency through Selective FL Algorithm), an innovative approach to client selection that enhances model performance while reducing client-server communication rounds. MESFLA implements a two-phase strategy:

1. Initial clustering phase based on data weight or model similarity, where client models are transmitted to the server and retained to prevent performance decay
2. Selective client participation within clusters based on relevance metrics

This approach significantly improves system efficiency by:

1. Implementing a 5% convergence threshold for global model updates

2. Eliminating unnecessary model transmissions in each round
3. Facilitating more accurate comparisons of accuracy and loss metrics
4. Reducing overall communication overhead

Experimental results validate MESFLA's effectiveness, demonstrating improved performance metrics across training rounds through its clustering-based client selection strategy. The approach of retaining global models at both client and server levels, while selectively transmitting updates, proves particularly effective for accelerating convergence.

Our research advances the field's understanding of both local update optimization and client selection strategies in FL systems. These contributions provide practical solutions to key challenges in FL, particularly in scenarios involving heterogeneous client capabilities and resource constraints. Future work could explore additional adaptation mechanisms and extend MESFLA's applicability to more diverse FL scenarios.

6.1 Limitations

Despite the significant contributions presented in this thesis, several limitations should be acknowledged to provide a comprehensive view of our research and identify opportunities for future improvements. While our approach addresses non-IID data challenges through clustering and selective client participation, the MESFLA framework was primarily tested on the MNIST dataset. This dataset, though widely used for benchmarking purposes, may not fully represent the extreme data heterogeneity encountered in real-world federated learning deployments. More complex datasets with greater feature dimensionality and class imbalance could present additional challenges to our clustering approach.

Our experimental evaluations were conducted with a limited number of clients (100) in a controlled environment. In large-scale federated learning deployments involving thousands or millions of clients, the computational overhead of CKA-based clustering might become prohibitive. The current implementation of MESFLA may require hierarchical or approximate clustering approaches to maintain efficiency at scale.

Our work primarily addresses data heterogeneity while making simplified assumptions about computational capabilities across clients. In practice, federated learning systems must contend with significant variations in processing power, memory constraints, and energy limitations across participating devices. MESFLA's client selection strategy does not explicitly account for these hardware constraints, which could lead to performance bottlenecks in real-world implementations.

The performance of MESFLA depends on several hyperparameters, including the convergence threshold and clustering parameters. Our evaluation provides limited sensitivity analysis for these parameters, and their optimal values may vary significantly

across different applications and datasets. Automatic hyperparameter tuning mechanisms would enhance the practical applicability of our approach.

6.2 Future Works

The local update is also one possible future work, as the FL needs to add on a training step. This needs to facilitate clients' faster training in a global model, as FEDALA shows his approach by using weights to determine how much training each client will initially do in the first round. Finally, the exploration of dynamic clustering strategies represents a significant frontier, particularly in developing adaptive mechanisms that can reconfigure client groupings in response to evolving data distributions and network conditions. Such approaches would need to balance clustering stability with responsiveness to system changes.

Perhaps most challenging is the creation of mobility-aware FL frameworks that can anticipate and adapt to client movement patterns. These frameworks would need to incorporate predictive mobility models, connection quality forecasting, and adaptive participation strategies to maintain learning continuity despite physical client mobility.

6.3 Published Works

Part of the results of this proposal is already publiched in a journal and workshops:

1. **BARROS, Alex;** ROSÁRIO, Denis; CERQUEIRA, Eduardo; FONSECA, Nelson L. S. da. A strategy to the reduction of communication overhead and overfitting in Federated Learning. In: *WORKSHOP DE GERÊNCIA E OPERAÇÃO DE REDES E SERVIÇOS (WGRS)*, 26. , 2021, Uberlândia. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2021 . p. 1-13. ISSN 2595-2722. DOI: https://doi.org/10.5753/wgrs.2021.17181.

2. **BARROS, A;** , Veiga, R., Morais, R., Rosário, D., Cerqueira, E. (2024). MES-FLA: Model Efficiency through Selective Federated Learning Algorithm. Journal of Internet Services and Applications, 15(1), 495–507. https://doi.org/10.5753/jisa.2024.4044

References

[1] A. Akhtarshenas, M. A. Vahedifar, N. Ayoobi, B. Maham, T. Alizadeh, S. Ebrahimi, and D. López-Pérez, "Federated learning: A cutting-edge survey of the latest advancements and applications," *Computer Communications*, vol. 228, p. 107964, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366424003116

[2] B. Yurdem, M. Kuzlu, M. K. Gullu, F. O. Catak, and M. Tabassum, "Federated learning: Overview, strategies, applications, tools and future directions," *Heliyon*, vol. 10, no. 19, p. e38137, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405844024141680

[3] 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.

[4] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," 2020.

[5] J. So, B. Guler, and A. S. Avestimehr, "Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning," 2021.

[6] F. Piccialli, D. Chiaro, P. Qi, V. Bellandi, and E. Damiani, "Federated and edge learning for large language models," *Information Fusion*, vol. 117, p. 102840, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253524006183

[7] A. Mukherjee and B. Rojas, "Business models for the long-term storage of internet of things use case data, idc report - market perspective,https://www.idc.com/getdoc.jsp?containerid=prap46737220," jul-2020.

[8] 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.

[9] P. Kairouz, H. B. McMahan, B. Avent, and et al., "Advances and open problems in federated learning," 2021.

[10] C. He, S. Li, J. So, X. Zeng, M. Zhang, H. Wang, X. Wang, P. Vepakomma, A. Singh, H. Qiu, X. Zhu, J. Wang, L. Shen, P. Zhao, Y. Kang, Y. Liu, R. Raskar, Q. Yang, M. Annavaram, and S. Avestimehr, "Fedml: A research library and benchmark for federated machine learning," 2020.

[11] J. Wang, Z. Charles, Z. Xu, G. Joshi, H. B. McMahan, B. A. y Arcas, M. Al-Shedivat, G. Andrew, S. Avestimehr, K. Daly, D. Data, S. Diggavi, H. Eichner, A. Gadhikar, Z. Garrett, A. M. Girgis, F. Hanzely, A. Hard, C. He, S. Horvath, Z. Huo, A. Ingerman, M. Jaggi, T. Javidi, P. Kairouz, S. Kale, S. P. Karimireddy, J. Konecny, S. Koyejo, T. Li, L. Liu, M. Mohri, H. Qi, S. J. Reddi, P. Richtarik, K. Singhal, V. Smith, M. Soltanolkotabi, W. Song, A. T. Suresh, S. U. Stich, A. Talwalkar, H. Wang, B. Woodworth, S. Wu, F. X. Yu, H. Yuan, M. Zaheer, M. Zhang, T. Zhang, C. Zheng, C. Zhu, and W. Zhu, "A field guide to federated optimization," 2021.

[12] NVIDIA, "Triaging covid-19 patients: 20 hospitals in 20 days build ai model that predicts oxygen needs," 2020.

[13] B. Liu, N. Lv, Y. Guo, and Y. Li, "Recent advances on federated learning: A systematic survey," 2023.

[14] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," 2018. [Online]. Available: https://arxiv.org/abs/1806.00582

[15] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," 2023.

[16] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, B. McMahan, T. Van Overveldt, D. Petrou, D. Ramage, and J. Roselander, "Towards federated learning at scale: System design," in *Proceedings of Machine Learning and Systems*, A. Talwalkar, V. Smith, and M. Zaharia, Eds., vol. 1, 2019, pp. 374–388. [Online]. Available: https://proceedings.mlsys.org/paper_files/paper/2019/file/bd686fd640be98efaae0091fa301e613-Paper.pdf

[17] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečný, S. Kumar, and H. B. McMahan, "Adaptive federated optimization," 2020.

[18] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," 2021.

[19] T.-M. H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," 2019.

[20] H. B. McMahan and M. Streeter, "Adaptive bound optimization for online convex optimization," 2010.

[21] S. Dutta, G. Joshi, S. Ghosh, P. Dube, and P. Nagpurkar, "Slow and stale gradients can win the race: Error-runtime trade-offs in distributed sgd," 2018.

[22] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," 2020.

[23] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, "Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization," 2020.

[24] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, "Federated learning with matched averaging," 2020.

[25] L. Gao, H. Fu, L. Li, Y. Chen, M. Xu, and C.-Z. Xu, "Feddc: Federated learning with non-iid data via local drift decoupling and correction," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 10 102–10 111.

[26] H. Nguyen, H. Warrier, and Y. Gupta, "A novel approach for federated learning with non-iid data," in *2022 9th International Conference on Soft Computing Machine Intelligence (ISCMI)*, 2022, pp. 62–67.

[27] T. K. Dang, X. Lan, J. Weng, and M. Feng, "Federated learning for electronic health records," *ACM Trans. Intell. Syst. Technol.*, vol. 13, no. 5, Jun. 2022. [Online]. Available: https://doi.org/10.1145/3514500

[28] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "Emnist: an extension of mnist to handwritten letters," 2017.

[29] T. T. F. Authors, "Tensorflow federated stack overflow dataset," 2019.

[30] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images." 2009.

[31] T.-M. H. Hsu, H. Qi, and M. Brown, "Federated visual classification with real-world data distribution," 2020.

[32] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečný, H. B. McMahan, V. Smith, and A. Talwalkar, "Leaf: A benchmark for federated settings," 2019.

[33] J. Luo, X. Wu, Y. Luo, A. Huang, Y. Huang, Y. Liu, and Q. Yang, "Real-world image datasets for federated learning," 2021.

[34] F. Sattler, K.-R. Müller, and W. Samek, "Clustered federated learning: Model-agnostic distributed multi-task optimization under privacy constraints," 2019.

[35] C. Briggs, Z. Fan, and P. Andras, "Federated learning with hierarchical clustering of local updates to improve training on non-iid data," 2020.

[36] Q. Jing, W. Wang, J. Zhang, H. Tian, and K. Chen, "Quantifying the performance of federated transfer learning," 2019.

[37] Y. Liu, Y. Kang, C. Xing, T. Chen, and Q. Yang, "A secure federated transfer learning framework," *IEEE Intelligent Systems*, vol. 35, no. 4, pp. 70–82, jul 2020. [Online]. Available: https://doi.org/10.1109%2Fmis.2020.2988525

[38] W. Liu, L. Chen, Y. Chen, and W. Zhang, "Accelerating federated learning via momentum gradient descent," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 8, pp. 1754–1766, 2020.

[39] Z. Jiang, A. Balu, C. Hegde, and S. Sarkar, "Collaborative deep learning in fixed topology networks," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 5906–5916.

[40] A. Koskela and A. Honkela, "Learning rate adaptation for federated and differentially private learning," 2019.

[41] H. Mostafa, "Robust federated learning through representation matching and adaptive hyper-parameters," 2019.

[42] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-iid data with reinforcement learning," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 2020, pp. 1698–1707.

[43] L. U. Khan, W. Saad, Z. Han, E. Hossain, and C. S. Hong, "Federated learning for internet of things: Recent advances, taxonomy, and open challenges," *IEEE Communications Surveys Tutorials*, vol. 23, no. 3, pp. 1759–1799, 2021.

[44] T. Zhang, L. Gao, C. He, M. Zhang, B. Krishnamachari, and A. S. Avestimehr, "Federated learning for the internet of things: Applications, challenges, and opportunities," *IEEE Internet of Things Magazine*, vol. 5, no. 1, pp. 24–29, 2022.

[45] 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.

[46] Z. Qu, R. Duan, L. Chen, J. Xu, Z. Lu, and Y. Liu, "Context-aware online client selection for hierarchical federated learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 12, pp. 4353–4367, 2022.

[47] Z. Qiao, Y. Shen, X. Yu, J. Zhang, S. Song, and K. B. Letaief, "Content-aware client selection for federated learning in wireless networks," in *2022 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*, 2022, pp. 49–54.

[48] D. B. Ami, K. Cohen, and Q. Zhao, "Client selection for generalization in accelerated federated learning: A multi-armed bandit approach," 2023.

[49] J. L. R. Sousa, W. Lobato, D. Rosário, E. Cerqueira, and L. A. Villas, "Entropy-based client selection mechanism for vehicular federated environments," in *Proceedings of the 22nd Workshop on Performance of Computer and Communication Systems*. SBC, 2023, pp. 37–48.

[50] 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*, 2021, pp. 54–66.

[51] 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, 2020.

[52] H. T. Nguyen, V. Sehwag, S. Hosseinalipour, C. G. Brinton, M. Chiang, and H. V. Poor, "Fast-convergent federated learning," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 201–218, 2020.

[53] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 19 586–19 597, 2020.

[54] A. Albaseer, M. Abdallah, A. Al-Fuqaha, and A. Erbad, "Client selection approach in support of clustered federated learning over wireless edge networks," in *2021 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2021, pp. 1–6.

[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*, pp. 1–15, 2023.

[56] Y. Fraboni, R. Vidal, L. Kameni, and M. Lorenzi, "Clustered sampling: Low-variance and improved representativity for clients selection in federated learning," in *International Conference on Machine Learning*. PMLR, 2021, pp. 3407–3416.

[57] C. Li, G. Li, and P. K. Varshney, "Federated learning with soft clustering," *IEEE Internet of Things Journal*, vol. 9, no. 10, pp. 7773–7782, 2022.

[58] J. R. R. Tolkien, *The Fellowship of the Ring: Being the first part of The Lord of the Rings*. England, UK: Houghton Mifflin Harcourt, 2012, vol. 1.

[59] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," 2017.

[60] X. Hu, R. Li, Y. Ning, K. Ota, and L. Wang, "A Data Sharing Scheme Based on Federated Learning in IoV," *IEEE Transactions on Vehicular Technology*, pp. 1–13, 2023.

[61] Y. Xiong, R. Wang, M. Cheng, F. Yu, and C.-J. Hsieh, "Feddm: Iterative distribution matching for communication-efficient federated learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 16 323–16 332.

[62] A. Barros, D. Rosário, E. Cerqueira, and N. Fonseca, "A strategy to the reduction of communication overhead and overfitting in federated learning," in *Proceedings of the 26th Workshop on Management and Operation of Networks and Service (WGRS)*. Porto Alegre, RS, Brasil: SBC, 2021, pp. 1–13.

[63] R. Veiga, C. Both, I. Medeiros, D. Rosário, and E. Cerqueira, "A federated learning approach for authentication and user identification based on behavioral biometrics," in *Proceedings of the 41st Brazilian Symposium on Computer Networks and Distributed Systems*. Porto Alegre, RS, Brasil: SBC, 2023, pp. 15–28. [Online]. Available: https://sol.sbc.org.br/index.php/sbrc/article/view/24526

[64] Y. Chen, X. Sun, and Y. Jin, "Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 10, pp. 4229–4238, 2020.

[65] C. Smestad and J. Li, "A systematic literature review on client selection in federated learning," in *Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering*, ser. EASE '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 2–11. [Online]. Available: https://doi.org/10.1145/3593434.3593438

[66] R. Song, L. Zhou, V. Lakshminarasimhan, A. Festag, and A. Knoll, "Federated Learning Framework Coping with Hierarchical Heterogeneity in Cooperative ITS," in *IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, Oct. 2022.

[67] W. Lobato, J. B. D. D. Costa, A. M. d. Souza, D. Rosário, C. Sommer, and L. A. Villas, "FLEXE: Investigating Federated Learning in Connected Autonomous Vehicle Simulations," in *IEEE 96th Vehicular Technology Conference (VTC-Fall)*. IEEE, Sep. 2022.

[68] S. Kornblith, M. Norouzi, H. Lee, and G. Hinton, "Similarity of neural network representations revisited," in *International conference on machine learning*. PMLR, 2019, pp. 3519–3529.

[69] 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

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