ADDIS ABABA UNIVERSITY
COLLEGE OF TECHNOLOGY AND BUILT ENVIRONMENT(CTBE)
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

MITIGATING ISSUES CAUSED BY DEVICE DROPOUT IN KNOWLEDGE DISTILLATION BASED FEDERATED LEARNING

BY
EYERUSALEM BANTE

ADVISOR
Dr. FITSUM ASSAMNEW

A thesis submitted to the School of Electrical and Computer Engineering in partial fulfillment of the requirements for the Degree of Master of Science in Computer Engineering

March, 2026
ADDIS ABABA, ETHIOPIA

ADDIS ABABA UNIVERSITY
COLLEGE OF TECHNOLOGY AND BUILT ENVIRONMENT(CTBE)
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

MITIGATING ISSUES CAUSED BY DEVICE DROPOUT IN KNOWLEDGE DISTILLATION BASED FEDERATED LEARNING

BY
EYERUSALEM BANTE

APPROVED BY BOARD OF EXAMINERS

Dr. Sosina Mengistu	_____	_____
Dean, SECE, CTBE	Date	Signature
Dr. Fitsum Assamnew	_____	_____
Advisor	Date	Signature
Dr. Biniam Tadesse	_____	_____
Internal Examiner	Date	Signature
Dr. Yonas Yehualaeshet	_____	_____
External Examiner	Date	Signature

Declaration

I, Eyerusalem Bante Kassie, declare that this research paper, titled ”Mitigating issues caused by device dropout in knowledge distillation based federated learning”, is my original work.All sources of information in this study have been properly acknowledged and cited. I also confirm that this work has not been previously submitted, in its entirety or in part, for any other acedemic qualification at this or any other institution.

Declared By:

_____

Student’s Name

_____

Signature

Approved By:

_____

Advisor’s Name

_____

Signature

March, 2026

Acknowledgments

First and foremost, I would like to express my gratitude to the Almighty God for his guidance, strength, and wisdom throughout the course of this research.

I am also grateful to my academic advisor Dr. Fitsum Assamnew, for his guidance, insightful feedback, and unwavering support. This research would not have been possible without his expertise.

Finally, I want to thank my family and friends for their constant support, understanding and patience. Their encouragement provided me with the motivation to complete this work.

Abstract

Federated learning (FL) is a technique that enables clients collaboratively train a machine learning model on distributed data residing on heterogeneous client devices while preserving data privacy. Recent studies suggest that applying knowledge distillation (KD) in federated learning have the potential to address challenges such as device behavior diversity, communication issues, accuracy bottleneck on resource constrained device, slow convergence and others faced by FL. However, in knowledge Distillation based Federated learning, collaborative training on edge devices is adversely impacted by device dropout and intermittent offline behavior resulting from limited resources and unstable connectivity, leading to reduced robustness and convergence on the global model. This research mitigates these issues, using a dropout resilient KD based FL system that couples data similarity-based client clustering with battery charge level -aware client selection strategy. The clients are grouped according to data distribution similarity of their local data distribution, that creates redundancy pool for compensating dropout by substitute client from same cluster. A battery aware algorithm prioritizes devices with sufficient energy levels, a strategy that reduces the probability of mid round dropouts, ensuring statistical data preservation, and improved global model stability and convergence. Experimental evaluations across MNIST dataset and dropout rates demonstrate improved global model accuracy in non-IID settings among heterogeneous devices. The findings contribute to advancing robust and efficient KD based FL frameworks suitable for resource-constrained environments, including mobile and IoT devices.

Keywords: Federated Learning, Knowledge Distillation, Client Dropout, Battery-Aware Client Selection, Client Clustering, Non-IID Data

Table of Contents

List of Figures

List of Tables

List of Acronyms

FL Federated Learning

KD Knowledge Distillation

$Non-IID$ Not Independent and Identically Distributed

IID Independent and Identically Distributed

SGD Stochastic Gradient Descent

DNN Deep Neural Network

ML Machine Learning

AI Artificial Intelligence

IoT Internet of Things

CPU Central Processing Unit

GPU Graphics Processing Unit

CNN Convolutional Neural Network

DL Deep Learning

$EAFL$ Energy aware Federated Learning

$FedAvg$ Federated Averaging

$FedGKT$ Federated Group Knowledge Transfer

GD Gradient Decent

Wh Watt-hour

$Wi-Fi$ Wireless fidelity

Chapter 1

Introduction

1.1 Background

Federated learning (FL) is a machine learning method in which many clients (e.g. mobile devices, IOT devices) collaboratively train a model under orchestration of a central server (e.g., service provider), while keeping the training data decentralized. It is a privacy-preserving way for collaborative model training in decentralized data sources and large-scale networks. Unlike centralized learning, FL enables edge devices to learn from local data without transmitting it to a central server. Its ability to leverage distributed and privacy sensitive data has led to applications in healthcare, finance, mobile communications, and the Internet of Things (IOT), establishing FL as a key research area for a secure and efficient machine learning system [1] [2] [3].

In federated learning, the scale of federation implies the number and type of participating devices or nodes involved in the training process, and the larger the scale of federation the more diversified the training data will be leading to better model performance. According to the scale of federation, the parties involved in the training process are divided into two categories 'cross device' and 'cross silos' [4] [5]. The study focuses on a "cross-device" setting that involves training a global model collaboratively across a large number of edge devices of individual users, those edge devices include smart phones, wearable, IOT devices with limited resources imposing challenges in terms of processing power, storage capacity, battery life that can impact the training speed and efficiency of the federated learning process.

Many methodologies have been developed to address core challenges of federated learning consisting of heterogeneous data, high communication costs, and system heterogeneity. Traditional federated learning strategies such as Federated Averaging (FedAvg) provide a straightforward way to aggregate model updates from the edge devices, assuming stable participation that is impractical in edge devices, struggling under non-IID (nonindependent and identically distributed) data or when there is edge device dropout leading to biased and inefficient model [6] [3] [7]. In FL, the negative impact of non-IID data distributions can be reduced through knowledge distillation-based training, which facilitates structured knowledge exchange between the central server and edge devices. Applying KD in FL several advantages in addressing key challenges, including improved learning under non-IID data distribution, enhanced generalization of the global model across diverse data and devices, and reduced communication overhead during model updates. Overall resulting in improved model performance, efficient deployment on edge devices and reduced training time by exchanging model outputs and/or model-agnostic intermediate representations instead of directly transferring model parameters/ model updates [3]. While the methods show promise, most studies fail to directly model client diversity specifically regarding non-IID data distributions, dynamic battery constraints and edge device availability.

.

Current solutions for client dropout in FL use oversimplified client selection by using random or greedy strategies that ignore dropped clients leading to update bias. Furthermore, many fail to integrate clustering or statistical insights to promote fair client representation [8]. Clustering-based methods improve the handling of non-IID data but rarely consider the joint dynamics of data similarity and device participation reliability. Other energy or battery aware protocols do not consider model or data diversity, while others also not integrating knowledge distillation [9] [10] [11]. The need for a solution that considers client data statistics, battery levels and knowledge transfer for resilient federated learning client participation remains.

Our research focus to address the limitations by integrating data distribution aware clustering with energy aware client selection, by effectively selecting clients balancing data representation, energy constraints and risk of client dropout. The system delivers models that are more representative and robust particularly in key applications such as health monitoring or smart city infrastructure where device resources and availability are unpredictable [12] [13]. On top of that incorporating knowledge distillation improves model personalization and learning stability under heterogeneous devices. Our strategy helps avoid repetitive selection of only energy rich clients by probabilistic client selection, enabling representative contribution across training rounds. This tackles client dropout issue and can enhance technical effectiveness of federated learning systems in real world.

This research aims to develop and validate knowledge distillation based federated learning system, consisting of cluster-based client dropping and battery-aware client selection. The main objectives are to mitigate the negative impacts of device dropout by: (1) transmitting client data distribution statistics for similarity-based clustering, (2) prioritizing client selection by cluster diversity and battery level awareness [9] [7] [14]. This approach expected to enhance resilient, efficient and generalized federated learning system, even if there are client availability fluctuations due to limited resources or environmental changes.

Here the study further explored the integration of FL and Knowledge distillation (KD)for resource efficient joint learning [15][7]. This chapter introduces the foundational concepts, reviews existing literature and advances in FL and KD, the outcome of deploying such systems on resource-constrained edge devices.

1.2 Problem Statement

Previous works integrating KD into the training process of FL have been successful in addressing issues of constrained device resources, customizing FL training process to heterogeneous devices, adapting to complex communication channels in the FL training as well as network typologies [16]. However, Frequent device dropouts in collaborative knowledge distillation-based federated learning system impact the training processes leading to incomplete knowledge transfer, increased global model training time, outdated local models, unequal workload distribution among clients which affect the model training performance, accuracy as well as global model convergence speed [17] [18] [19] [6]. Since it is impractical to keep devices online all the time, how to cope with device dropout in knowledge distillation based federated learning systems remains a challenge [16].

In our proposed work, the focus is how knowledge can be retained in the system despite participating devices dropping out. We propose a method to select nodes that are capable of restoring data from at risk or nodes with drop out history in the system. Some nodes can be selected as representatives of other clients, having enough resource to carry out the training process, having a history of consistent participation or good network connectivity. We ask the following questions to arrive at the appropriate solution.

RQ1 Which client devices traits is most effective for selection mechanisms to minimize dropout impact in Knowledge distillation-based federated learning systems?

RQ2 How can trait-based client selection strategies retain acceptable model performance and convergence in KD-FL systems under frequent edge device dropouts?

1.3 Objectives

General Objective

Develop a method for knowledge distillation-based federated learning systems that mitigates the negative effects of frequent device dropouts, ensuring efficient knowledge transfer and improved model performance.

Specific Objectives

- To design and Implement a Dropout-Resilient Knowledge Distillation based Federated learning system utilizing client clustering based battery aware client selection Strategy.
- To develop a client selection mechanism for Quantifying and Adapting to Client Dropout during federated training.
- To evaluate the Efficiency of Knowledge Transfer under Varying Dropout Scenarios on the global model accuracy.
- To assess the Impact of the Proposed Method on Global Model accuracy and Convergence speed towards optimal performance.

1.4 Contribution

This study presents a novel approach to enhancing the robustness and efficiency of knowledge distillation-based federated learning systems by explicitly addressing the challenges posed by device dropout. The main contributions of this work are summarized as follows:

1. **Cluster-based Redundancy Strategy**: A client clusterinf strategy based on data distribution similarity is proposed to ensure data diversity while providing redundency during training. when device dropout occurs, clients with in the same cluster are leveraged to compensate for missing knowledge contributions, there by preserving the statistical representativeness of the training sata and improving the robustness of the global model.
2. **Battery-aware Client Selection**: Our algorithm prioritizes clients with sufficient battery levels from each cluster to minimize dropout risk. It also ensures fair participation across device categories (low-end, mid-range, high-end), balancing reliability with comprehensive data representation.

1.5 Scope and Limitation

1.5.1 Scope

This research enhances the robustness of KD based FL systems in resource constrained environments by addressing client dropout.

The client clustering mechanism groups clients by statistical data distribution similarity, enabling coherent replacement from the same cluster during dropouts. Battery level aware client selection prioritizes energy sufficient clients while maintaining data diversity through controlled sufficient battery participation. The system employs split learning architecture for the KD based FL frame work, in which clients transmit extracted features and logits to the server for classification and feature based knowledge distillation.

The proposed system is evaluated through simulations on non-IID datasets with synthetic battery profiles to model energy dynamics. Its performance is bench marked against state of the art feature based KD FL system with client clustering applied on it through the metrics such as global model accuracy, dropout resilience, convergence speed, and resource efficiency.

1.5.2 Limitation

This study has several limitations. First, it assumes access to client level data distribution statistics, which may not always be feasible in strict privacy preserving scenario. The client replacement mechanisms may not fully recover knowledge from clients with highly unique or non-redundant data distributions. The system primarily addresses client dropout due to battery constraints and does not model other potential causes, such as unstable network connections, system failures, or user initiated interruptions. Finally, the framework focuses on dropout mitigation and does not explicitly address other FL challenges, including adversarial behavior, privacy leakage, or communication overhead, which remain areas for future work.

1.6 Organization of the study

This paper is structured as follows. Chapter two contains; theoretical background and literature review of federated learning and knowledge distillation based federated learning systems. Chapter three presents the elaborated methodology of data partitioning among federated clients, data distribution based client clustering and, client device battery modeling. The experimental results and discussion part of the study are presented in Chapter four, whereas Chapter five depict conclusion and future works regarding the research.

Chapter 2

Background and Literature Review

This study begins by establishing the foundation of federated learning (FL) as a decentralized machine learning approach that enables collaboration across client devices. Despite its advantages in scalability and reduced communication compared to centralized methods, FL encounters fundamental challenges such as client heterogeneity, resource constraints. Addressing these issues has led to various optimization techniques, among those the study focuses on knowledge distillation (KD) integration. However, persistent problems such as device dropout, especially in KD based FL where knowledge transfer depends on continuous client dropout remain inadequately solved.

2.1 Background

2.1.1 Federated learning

Federated learning is a decentralized machine learning approach that enables multiple clients (e.g., mobile phones, edge devices) to collaboratively train a shared global model under coordination of central server while keeping their local data private. This paradigm improves data privacy by reducing need for centralized data collection [15] [3]. There is no need to transfer sensitive raw data to a server, clients collaboratively train the shared model by periodically sending local model updates (weights or gradients) to a central server, that aggregates the updates using algorithms like Federated Averaging (FedAvg) [20]. FL is suitable for sensitive applications such as healthcare and finance; however, it introduces practical challenges like communication overhead, data heterogeneity and client resource variability and reliability.

A typical centralized FL system contains a central server which act as coordinator and distributed edge devices that participate in the training process. An iterative process of FL model training consists of four basic steps (1) All clients send locally computed gradients to the central server, (2) update the global model on the central server based on the gradients aggregated from the clients, (3) send the updated global model to the clients that will further participate in the training process, (4) the clients process the global model on their local data. This iteration will continue for a required amount of time that is until better model accuracy is found, or overall model convergence is better. Local models on devices like smartphones train on their local data, sending updates to a central server then, the server aggregates updates from various devices to create a global model.

In federated learning scale of federation implies the number and type of participating devices or nodes involved in the training process and the larger the scale of federation the more diversified the training data will be leading to better model performance. Under the scale of federation, the parties that are involved in the training process are divided into two categories “cross-device” and “cross-silo” [4] [5]. The study focuses on “cross-device” setting that involves training a global model collaboratively across large number of individual user edge devices those edge devices include, smart phones, wearable, IOT devices with limited resources imposing challenges in terms of processing power, storage capacity, battery life that can impact the training speed and efficiency of the federated learning process.

In cross-device setting, users(clients) interact with their devices in unique ways leading to skewed data distributions besides the FL training data of each user depends largely on the use of specific local devices. These heterogeneous data or what is also called non-IID data (not Independent and Identically Distributed data) impose challenge to achieve edge intelligence in FL [5] [21] . Non-IID data refers to scenarios where clients hold data samples drawn from different probability distributions, violating the IID assumption of traditional machine leaning. FL models that have been trained on non-IID data distribution have an impact on the global model performance due to its biased nature, which performs well on data from most participants but poorly on data from the minority. Additionally, if the data from individual clients in the training are far from the global average it takes longer time to converge. This issue of non-IID data and its consequences in the training process of FL can be mitigated by Knowledge Distillation (KD) which involves sharing knowledge between the teacher model and the edge devices.

2.1.2 Knowledge distillation

Knowledge distillation (KD) is a training technique where a smaller "student" model learns to mimic the outputs such as logits or intermediate representations of larger or ensemble "teacher" model [22] [3]. The objective is to transfer knowledge from a complex teacher model to a simpler student model while preserving performance. Essentially, it is a form of model compression that was first successfully demonstrated by Bucilua and collaborators in 2006. It is a technique in deep learning that compresses large complex(teacher) model into smaller simpler model(student). This process of learning a small model from a larger model was formalized as a "Knowledge Distillation" [23]. The goal of KD is to have student model with performance similar or grater to that of the teacher model.

Applying KD in FL offers several advantages including, enhanced execution on non-IID data distribution, developing generalized model that can be applied on wider range of data, reducing communication overhead during an update. Overall resulting in improved model performance, efficient deployment on edge devices and reduced training time. KD based strategies can be applied to decouple client training from global model synchronization, allowing heterogeneous models and reducing communication overhead of the FL process by exchanging model outputs and/or model-agnostic intermediate representations instead of directly transferring model parameters/model updates. Studies show that KD proves to be valuable tool in FL training of efficient and accurate models while addressing the specific challenges of FL [3]. This is especially practical in scenarios involving clients with limited computation resources, as it allows distilled knowledge (e.g., logits or features) to be used without sharing full model parameters.

The application of knowledge distillation (KD) in FL can be categorized into variety of application paradigms. The first approach centralized KD (serevr side KD) which is the most common, has an advantage to handle model heterogeneity and communication costs. Clients train their local models on private data, then send their model's logits/soft labels to the server and the server aggregates these soft labels and uses them to train larger "teacher" model which provides guidance to student models (the clients) [24] [1] . The second one is decentralized KD, that makes distillation process between clients allowing them to learn from each other without a central aggregator to improve personalization and reduce reliance on single central server [25]. The third approach is split learning-based KD, which is powerful variant were model is split between client and server.

In split learning, which is collaborative model training, clients train portion of the neural network (e.g., feature extractor) and send the intermediate features to the server then the server completes the forward and backward passes. Here KD is applied by having the server's "teacher" model provide guidance to clients' "student" feature extractors [26] [14]. The third approach is the one that is used for this research.

2.1.3 Feature-based KD and split learning

Knowledge distillation-based FL systems can be categorized based on type of knowledge that is transferred between the models. Response based approach is most common that involves student model mimicking the output logits or soft labels of teacher [27].

Feature based distillation, transfers knowledge from the intermediate layers of a neural network [1]. Relation-based distillation is more advanced category, focusing on transferring the relationships between data points or features [3]. Here on our study, we are focusing on feature-based distillation.

Feature-based knowledge distillation extends traditional KD by clients sending extracted intermediate feature representations in and logits to the server, rather than model updates [7]. When combined with split learning, where the model is splited between server and clients, clients perform partial forward passes, and the server completes the model training. This split learning strategy reduces client limitation problem in heterogeneous resource-constrained FL environments, while preserving detailed knowledge flows. It is especially effective in scenarios with frequent client dropout [28].

2.1.4 Federated optimization techniques

Federated learning (FL) as compared to centralized machine learning, has unique challenges for optimization due to decentralized data storage, communication constraints, and client heterogeneity. FL systems must handle non-IID data distributions, dynamic client participation, and frequent device dropout all of which degrade model performance. To address these issues, researchers have developed a verity of federated optimization techniques. These can broadly be categorized in to: (1) synchronous averaging algorithms, (2) communication efficient and model compression methods, (3) personalization methods, (4) variance reduced and adaptive techniques [15].

These are adopted to improve according to target application and system constraints, to improve training stability, convergence speed fairness, and model generalizability across heterogeneous devices and data distributions.

Synchronous averaging algorithm is an optimization method where the central server waits for set of clients to complete their local training before aggregating their update. One of these is Federated averaging (FedAvg), that performs well under IID data settings but suffers under non-IID conditions, also its efficiency degrade when clients vary in size and computation speed [20]. Communication efficiency is the ability to reduce the amount of data transmitted between central server and client devices during training. It is major bottleneck in FL, particularly in bandwidth constrained and large-scale environments. Gradient sparcification, quantization, pruning, local update compression and Knowledge distillation (KD) are methods that reduce update size exchanged between clients and server which is crucial for bandwidth-limited environments [29].

Real world federated clients have non-IID data, optimization technique such as personalized FL have been proposed. This approach allows each client to fine-tune the global model to its local data. Works like pFed-Me (personalized FL based on model enhancement) apply frameworks so that clients can adopt the global model with minimal local data [30]. Variance-reduced and adaptive techniques aim to reduce the "noise" or high variance in gradient updates, leading to more stable and faster model convergence. Frameworks like Federated optimization in heterogeneous networks (Fedprox) and scaffold are one of the techniques which address specifically local update drift [31] [32] [33].

2.2 Related works

2.2.1 Federated learning and its fundamental challenges

Federated Learning (FL) is a concept first introduced by Google in 2016, in which multiple devices collaboratively learn a machine learning model without sharing their private data under the supervision of central server [34]. This offers ample opportunities in critical domains such as healthcare, finance etc, where it is risky to share private user information to other organization or devices.

Similarly, another review paper states that unlike centralized machine learning model training, federated learning is a distributed machine learning process, which allows multiple nodes to work together to train a shared model without exchanging raw data. It offers several key advantages, such as data privacy, security, efficiency, and scalability, by keeping data local and only exchanging model updates through the communication network [35].

Client heterogeneity is differences among participating devices in federated learning in their local data distribution (statistical heterogeneity), computational power, communication capacities and participation behaviours, that is because when FL systems deployed in real world environments. One major source of heterogeneity arises from clients generating or possessing data with diverse characteristics such as varying class distributions, differing sample sizes, or domain specific features leading to non-independent and identically distributed (non-IID) data across the devices, where clients collect data specific to their usage patterns, environments, or regions, leading to the statistical heterogeneity [36] [8] . This data heterogeneity often results in "drift" between the local and the global model objectives, impede the global model's ability to generalize which slows down convergence particularly as the number of clients increases [37]. Data heterogeneity issues emerge from the discrepancies in data distributions across various devices in the distributed networks. this variance is quite common in real-world scenarios, often due to the diversity of users' behaviors or performances. This heterogeneity also degrades model convergence and accuracy if not handled properly, it also leads to in consistent client participation, training delays and even causes client dropout [15]. Another one is behavioral heterogeneity, in which the frequency and duration of client availability introduce in consistency in client participation across rounds [4]. Such inconsistency poses challenges for fair and efficient training, along with maintain model convergence.

Resource constraint is another challenge in FL, primarily as FL deployed on edge or IoT devices that are battery-powered with limited hardware capabilities, computational power, battery life, storage and unreliable communication channels [38][39] [40]. These constraints affect the ability of clients to perform local training or participate reliably across multiple communication rounds. For instance, clients with limited battery led to device dropout, disrupting collaborative learning and negatively impacting model convergence and performance [11].

Despite the expensive research on FL, deploying FL in practical application introduces some bottlenecks, which affects the performance and efficiency of FL model. As noted by Jie Wen et al. (2023) [38], communication overhead is major bottleneck for FL, because the communication cost is greater than the computation cost when numerous edge devices are sending their model parameters to central server. Moreover, device heterogeneity is another challenge.

In the process of aggregation, the federated device heterogeneity problems in practical applications, data heterogeneity arises from non-IID nature of client data distributions, while model heterogeneity result from differences in device resource capacities, model architectures and personalized task. These heterogeneity problems may reduce the performance of the global model or makes it difficult to get convergence to some extent. To address these issues, Jie Wen et al., propose strategies such as model compression and client selection, which aim to reduce the number of communication rounds and scale of parameter Transmissions, thereby improving communication efficacy and accelerating convergence. Additionally, in order to alleviate heterogeneity Challenge they have focused on solutions such as device scheduling, meta learning, and transfer learning.

While Jie Wen et al. primarily focuses on communication and heterogeneity challenges, Tuo Zhang et al. 2022 [41] emphasizes the resource limitation of edge devices. Existing ML models, especially deep neural networks, are known to be computation intensive, which presents strict requirements on hardware and may result in low training efficiency on edge devices. As a mitigation strategy, they propose the development of customized and specialized hardware for machine learning applications on the edge is a promising direction to accelerate inference and training tasks while using much less energy compared to general-purpose processors.

Moreover, bandwidth limitations and high communication overhead due to model transmission further increase training latency and energy usage [42]. Such resource constraints may force FL systems trade-off between diverse population of weak participants (coverage) or smaller subset of high-capacity clients (efficiency). Thus, addressing resource constraints is critical for sustainable, inclusive, and efficient FL deployments.

Integrating knowledge distillation in to federated learning can address several key challenges by reducing communication overhead, mitigating model heterogeneity, and enabling collaboration between heterogeneous models through soft knowledge transfer rater than direct weight sharing [43] [44].

2.2.2 Knowledge distillation in Federated learning

Knowledge distillation refers to the process of transferring the knowledge from a large unwieldy model or set of models to a single smaller model that can be practically deployed under real world constraints. Essentially, it is a form of model compression that was first successfully demonstrated by Bucila and collaborators in 2006 [45]. According to Xiaoyi pang et al. (2024) [46], knowledge distillation (KD) a technique for transferring learned knowledge from one network to another in a light weight and efficient manner, can be incorporated in to FL to address its inherent limitation as well as to enhance its usability and universality in IoT. KD based FL can effectively achieve collaborative learning among resource limited devices that are heterogeneous in data distribution, model architectures, or quality of resources and offer enhanced privacy guarantees.

Knowledge distillation has emerged as a dynamic technique to address challenges in FL. In KD, instead of sharing entire models, client and servers exchange "soft labels" (output logits of model) or intermediate feature representation of a model [3] [22]. KD is integrated into FL primarily to overcome limitations of standard parameter-based averaging algorithms such as Federated averaging (FedAvg). The key challenges of FL mitigated by KD include, data heterogeneity, model heterogeneity, communication-efficiency and privacy preservation.

Early works on KD in FL, "Distilling knowledge in neural network" and "Multi model federated learning" has been foundational for all subsequent works to mitigate issues in FL [47] [48]. Other KD approaches for FL have also been done for addressing data and model heterogeneity among clients, were devices with non-IID data and different model architecture handled by KD [49]. Methods such as FedDG and FedGKT, utilize KD into FL to enable clients to send features or logits instead of model weights, to address issues specifically related to model heterogeneity, resource constraints and communication overhead [50] [14]. The application of KD in FL have proven success in improving model performance while accommodating diverse device capabilities, but still vulnerable to dropout as they rely on consistent client participation.

Conversely, Laiqiao qin et al. (2024) [7] has stated, KD based solutions to FL challenges can broadly categorized into three approaches: (1) feature-based federated distillation, which transmits only the model's features between clients and the server, such as logits, attention, and intermediate features.

Since these features are much smaller and contain far less sensitive information than model parameters, they can effectively address the communication bottleneck and privacy risks in FL. (2) parameter-based federated distillation, which involve sharing model parameters between clients and the server. This method is primarily used to address the issue of model performance degradation caused by directly aggregating model parameters in non-IID scenarios. (3) data-based federated distillation, which requires clients to employ dataset distillation methods to compress their data into a small-scale dataset which is not widely adopted in FL. Focusing on the feature-based category, Chuanguang Yangl et al. (2023) [51], highlight that it leverages intermediate feature information to provide richer and more comprehensive supervisory signals. This involves transforming intermediate feature maps into meaningful knowledge and employing algorithms that guide the student model to better replicate teacher behavior, including distillation methods tailored for vision transformers. Feature based KD emphasizes using intermediate-level information to guide the student network, addressing the limitations of response-based KD by providing more comprehensive supervision.

Split learning (SL) and federated learning (FL) are both model to data distributed machine learning paradigm that enable training without sharing raw data. Both FL and SL have their strength and drawbacks in model training. According to the findings of Chandra Thapa et al. (2022) [52], split fed is proposed as a hybrid framework that integrates the strengths of FL and SL. It combines FL'S parallelism with SL's enhanced privacy protection, which is also more suitable for resource constrained devices, while mitigating the limitations of each approach. Similarly, Chaoyang He et al. 2020 [14] investigate a framework that combines the strengths of federated learning, knowledge distillation, and split learning. It enables resource-efficient training on edge devices by having small models learn from a large server model via KD, reducing local computation and communication. It addresses the challenge of training large CNNs on resource-constrained edge devices. Scaling up the convolutional neural network (CNN) size (e.g., width, depth, etc.) is known to effectively improve model accuracy. However, the large model size impedes training on resource-constrained edge devices. For instance, federated learning (FL) may place undue burden on the compute capability of edge nodes, even though there is a strong practical need for FL due to its privacy and confidentiality properties.

Small edge models train locally and periodically transfer knowledge to server via KD, while the server updates a large CNN. The process involves alternating optimization, with models sharing hidden features rather than entire weights, reducing communication and computation cost. By integrating FL's decentralization, SL's model partitioning, and KD's knowledge transfer, FedGKT experimental results on multiple datasets show comparable model accuracy to traditional FL methods but with significantly lower computational and bandwidth costs (lower resource requirements).

2.2.3 Client selection in Federated Learning : strategies and challenges

In federated learning, client selection is the process of choosing a subset of clients to participate in each training round. It is also crucial component for FL optimization, as not all participating clients will be available, reliable, or useful at any given time [53] [54]. Client selection strategies play a vital role by determining efficiency, accuracy, and fairness of the global model training process. Effective client selection is necessary to address challenges posed by clients such as device heterogeneity, as the server typically engages only subset of clients per round due to communication bandwidth, computational constraints, and client availability. Much research has explored diverse selection criteria and optimization methods to improve FL outcomes. Traditional approaches relay on random or availability-based selection, but more recent works incorporate client data quality, resource status, performance awareness, network conditions and fairness metrics to guide intelligent and adaptive selection [55].

Here is a research synthesis of diverse strategies, highlighting approaches that balance performance, fairness, energy efficiency, and adaptability in heterogeneous and constrained environments. Fairness aware selection recognizes the risk of under representation of clients with unique or minority data and incorporates fairness into client selection, balancing accuracy with equitable data representation among clients [56] [57]. Energy availability is also critical constraint, particularly for mobile and IoT devices, several strategies prioritize clients with sufficient battery levels to reduce client dropout and wasted computation so that to minimize overall energy consumption [11] [58] [59]. Also, other studies, focus on clients with higher local loss or training potential, to accelerate convergence with minimal communication overhead [60].

Learning and hierarchical based selection, seek to balance trade-off between latency, energy consumption, and model performance, to select clients under budget constraints, aiming to maximize successful participation [61] [55]. Another principal approach uses data distribution similarity to guide selection, grouping similar clients dynamically to reduce communication overhead and dropout rates. Such similarity based and clustering approaches improve convergence speed, reduce communication overhead and enhance model performance [62] [63] [64].

A recent study conducted by Lei Fu et al. (2023) [65] highlights that, in typical FL scenarios, clients often exhibit significant heterogeneity in terms of data distribution and hardware configurations. Thus, randomly sampling clients in each training round may not fully exploit the local updates from heterogenous clients, resulting in lower model accuracy, slower convergence rate, degraded fairness, etc. To tackle the FL client heterogeneity problem, various client selection algorithms have been developed. FL client selection (a.k.a. participant selection or device sampling) decides which client devices are chosen in each training round. An effective FL client selection scheme can significantly improve model accuracy, enhance fairness, strengthen robustness, and reduce training overheads.

Carl Smestad and Jingyue Li (2023) [66] reviewed various client selection strategies in federated learning, highlighting that the choice of approach depends on the specific challenges a study aims to address, such as heterogeneity, resource allocation, communication costa, and fairness. Common strategies include homogeneous dataset selection, where clients are chosen to create a more uniform combined dataset, clustered federated learning (CFL), which balances non-IID data while scheduling clients based on round latency and bandwidth. Other approaches involve resource condition-based selection that considers device computational and network capabilities, energy consumption minimization to maximize client participation while reducing power usage, and fairness-guaranteed selection techniques aim to promote equitable participation among clients, though often at the expense of training efficiency.

Clustered Federated Learning (CFL) was introduced as an efficient scheme to balance out the non-IID data problem. In order to address the issue of label distribution skew, a method that check the similarity between the aggregated data distribution of the selected clients and compare it to the global data distribution. There is also another approach to clustering clients by grouping them according to classes of data and then randomly selecting one client with in every group.

Conversely, a separate technique has introduced diversity into client selection by measuring how a subset of clients can represent the whole when aggregated on the server for each communication round. Yousef Alsenani et al. (2024) [67] introduced FedSiKD, which integrates knowledge distillation (KD) in to a similarity based federated learning framework. In their approach, clients securely share relevant statistics about their local data distribution upon joining the system, enabling the formation of clusters with intra-cluster homogeneity. This enhances optimization efficiency and accelerates the learning process, effectively transferring knowledge between teacher and student models and addressing device constraints.

Amna Arouj and Ahmed M. Abdekmoniem (2022) [11] conducted a research study and developed a framework named EAFL, an energy-aware federated learning framework designed to address the limitations of existing methods by incorporating energy consumption considerations on battery-powered devices. The authors identify that traditional FL approaches often ignore device energy constraints, which can lead to high dropout rates and decreased model performance. To address this issue, they propose a novel client selection strategy that prioritizes devices with higher remaining battery levels, thereby maximizing participation and reducing energy drain. The methodology involves modeling the energy consumption of mobile devices and integrating this in to the selection process to balance training efficiency and energy usage. Experimental results show the model accuracy significantly improves and reduces the client device dropout rate up to 2.45 times compared to state of the art.

Client clustering based client selection in federated learning

Client clustering has emerged as a powerful strategy to counteract statistical heterogeneity caused by non-IID data across clients and enhance model performance. Clustering clients based on similarity is their data distribution or learned model representations allows grouping with homogeneous characteristics. These clusters are especially relevant in heterogeneous environments where a single model may not adequately represent diverse client populations, so clustering enable more refined aggregation, improving accuracy and convergence speed [68] [69]. There are many forms of client clustering in FL such as, hierarchical cluster formation to soft and adaptive methods each customized to improve model convergence, personalization, and resilience to non-IID data.

There are key clustering approaches in federated learning, that group clients based on data distribution or resource constraints, aiming to improve model performance and training efficiency. A hierarchical clustering technique that groups clients based on the similarity of their local update and training model per each cluster, giving better convergence and accuracy under heterogeneous data distribution among clients [70].

Other approaches, analyze correlation between model weights and data overlap of clients, to cluster clients enabling adaptive cluster formation while lowering computational demand and communication cost [71]. Similarity between model weights to cluster clients, effectively capturing both feature and label heterogeneity in clients, has been done by improving by demonstrating up 2% - 4% accuracy improvement with least resource overhead [72] [73] [74].

2.2.4 Client dropout challenge in Knowledge Distillation based Federated learning systems

Client dropout occurs when certain devices fail to participation a training round after being selected due to factors such as, battery depletion, network instability or computational limitations. Federated learning system depends on active and reliable participation of clients for consistent update, so dropout leads to biased global updates and degradation in accuracy and convergence speed. Simple approaches such as aggregating only available client updates may disproportionally exclude clients with unique data and favour overrepresented data. Advanced approaches like, Mimic and FL-FDMS mitigate client dropout by using similar client update substitution and also by adjusting received updates to reduce divergence induced by dropout [18] [6]. Strategies such as asynchronous updates, decentralized FL or redundant transmissions further increased communication overhead and compromise privacy. Therefore, client dropout remains a fundamental bottleneck to scalable, reliable FL deployments.

In knowledge distillation-based FL systems, where clients transmit intermediate features or logits instead of full model weights, client dropout introduces additional complexities. The knowledge distillation process relies heavily on continuous and representative feature and logit exchange. When clients' dropout frequently, knowledge transfer is incomplete and global learning slows, further degrading accuracy and convergence speed. The aggregated knowledge becomes less representative due to missing client updates, particularly in non-IID data settings [7].

Traditional KD based FL methods assume reliable client availability and presuppose availability of shared dataset or consistent model architecture across client. Thus, handling dropout in such systems remains open and critical area [22] [75].

Heqiang Wang et al. (2023) [18], investigated the phenomenon of client dropout in federated learning (FL), defined as the unexpected inactivity or withdrawal of participating clients during training due to factors such as resource constraints, unstable network connectivity, hardware failures, or user-driven interruptions. Unlike controlled client selection or sampling, dropout is typically uncontrollable and can occur unpredictably, adversely affecting FL convergence and performance. The authors identify the primary causes of dropout as stochastic and external, including hardware limitations, connectivity disruptions, and human intervention. Such dropouts can result in incomplete participation in training rounds, introducing bias in global model updates this in turn, can slow convergence, degrade accuracy, and reduce the overall robustness of FL models.

To address these challenges, the study proposes a friend model substitution strategy, wherein "friend" clients those with similar data distributions are dynamically identified based on pairwise similarity scores. When a dropout occurs, the missing updates are replaced by updates from these friend clients, thereby reducing substitution error and maintaining model quality.

Similarly, Yuchang Sun et al. (2024) [6] when federated learning (FL), being deployed at mobile edge networks, clients may have unpredictable availability and drop out of the training process, which hinders the convergence of FL. Here the client dropout was investigated to often caused by unpredictable availability due to factors such as limited battery power, poor network connectivity, or complete disconnection from the server. These dropouts introduce challenges, including oscillations around a stationary point of the global loss function, which occur became of the divergence between the aggregated update and the ideal central update causing convergence rate and training efficiency reduction. To address this their algorithm modifies each received model update based on historical updates, mimicking an imaginary central update regardless of the number of dropout clients. A correction variable is employed to adjust local updates, ensuring that the aggregated update aligns more closely with the central update and thus optimizes the global model learning process.

There is growing body of literature that explores about client dropout in federated learning according to the finding of Ignacy Stępka et al. (2025) [76] consider the problem of persistent client dropout in asynchronous Decentralized Federated Learning (DFL). Asynchronicity and decentralization obfuscate information about model updates among federation peers, making recovery from a client dropout difficult. Access to the number of learning epochs, data distributions, and all the information necessary to precisely reconstruct the missing neighbor's loss functions is limited.

Here there is an issue of persistent client dropout where clients become permanently unavailable during training and continues training without acknowledging dropout and forget the dropped client, where the client is removed from the federation and excluded from future updates, leading to loss of valuable data contribution. They have investigated adaptive strategy focusing on synthetic data reconstruction to replace the missing client. They have explored two main methods gradient inversion, which recovers data whose gradients align with those of the lost client, and model inversion, which assumes that the last available model is near a stationary point for its local objective and generated synthetic data accordingly.

In contrast to the above studies Zhiyuan Wu et al. (2024) [16] there are many open problems in KD-based FEL. In terms of device connectivity, it is impractical to keep devices online all the time, and how to cope with devices offline and dropout in KD-based FEL systems remains unsolved. This issue underscores the need for robust system designs capable of tolerating irregular device availability. The study further highlights the importance of developing dropout aware aggregation techniques, knowledge transfer protocols, and predictive models for device availability to proactively manage participation in training rounds.

Overall, in terms of device connectivity, it is impractical to keep devices online all the time, and how to cope with devices offline and dropout in KD based FEL remains unsolved.

Hassan Salman et al. (2025) [22] presents a case study in the healthcare domain, introducing a framework called FedHealth 2, a weighted federated transfer learning method that integrates batch normalization and knowledge distillation process, favoring certain clients. They have found that client dropout caused by factors such as limited battery life or unstable network connectivity can disrupt global model stability, emphasizing the need for strategies that ensure robust and reliable model training under real-world constraints.

Taken together, the literature indicates that there is a gap in addressing the impacts of client dropout in FL system even though there are works that has been done they still have some limitation, but our focus is mainly on KD based FL system. There is no prior work that is done to mitigate the problem of client dropout in those systems.

2.3 Summery

knowledge distillation (KD) has been effectively integrated with federated learning (FL) to address FL issues of device heterogeneity and resource limitation. But how to cope with frequent device dropouts remain challenge. so, we focus on retaining knowledge in the system despite client dropouts by employing a client selection mechanism to mitigate the negative impact of dropouts on system performance and also system stability. here we address the limitations by integrating data distribution aware clustering with energy aware client selection.

knowledge distillation further boosts model personalization and learning stability across heterogeneous devices, while the client selection prevents over reliance on energy rich clients, promoting fair contributions over training rounds. This yields robust, representative models for real-world applications like health monitoring and smart city infra structure, where device availability varies.

The study aims to develop and validate knowledge distillation based federated learning system, consisting of cluster-based client dropping and battery-aware client selection. The main objectives are to mitigate the negative impacts of device dropout by: (1) transmitting client data distribution statistics for similarity-based clustering, (2) prioritizing client selection by cluster diversity and battery level awareness [13] [9] [7]. This approach expected to enhance resilient, efficient and generalized federated learning system, even if there are client availability fluctuations due to limited resources or environmental changes.

Chapter 3

Methodology

3.1 Research Design

This chapter details the research approach, data preparation, proposed core methodological components, and training process of our core KD based FL system. The study adopts an experimental research design implementation and evaluating the KD based FL system performance. It integrates data distribution-based client clustering and battery level-aware probabilistic client selection in which the primary goal is to reduce the impact of client dropout, particularly in resource constrained environment. The research is motivated by real world challenges of client heterogeneity, non-IID data distributions and device energy limitations in FL. Our system adopts a centralized federated learning architecture with asynchronous updating mechanism [77]. In this setup, a single central server coordinates the training process, aggregating model update from selected heterogeneous clients at the end of each communication round. The system explicitly models client energy dynamics and non-uniform availability, focusing on the effects of mid training client dropouts on the system performance in terms of global model accuracy, robustness and fairness. Below on Figure 3.1 we can see the system architectural design for our proposed system.

3.2 Dataset preparation

3.2.1 Experimental Datasets

We employ an open-source widely adopted dataset particularly MNIST, due to their wide adoption in federated setups and its simplicity, from an official repository that ensures reproducibility. MNIST consists of 70,000 Gray scale images of handwritten digit split into 60,000 training images and 10,000 test images of 10 classes corresponding to digit 0-9, each being 28 x 28 pixels.

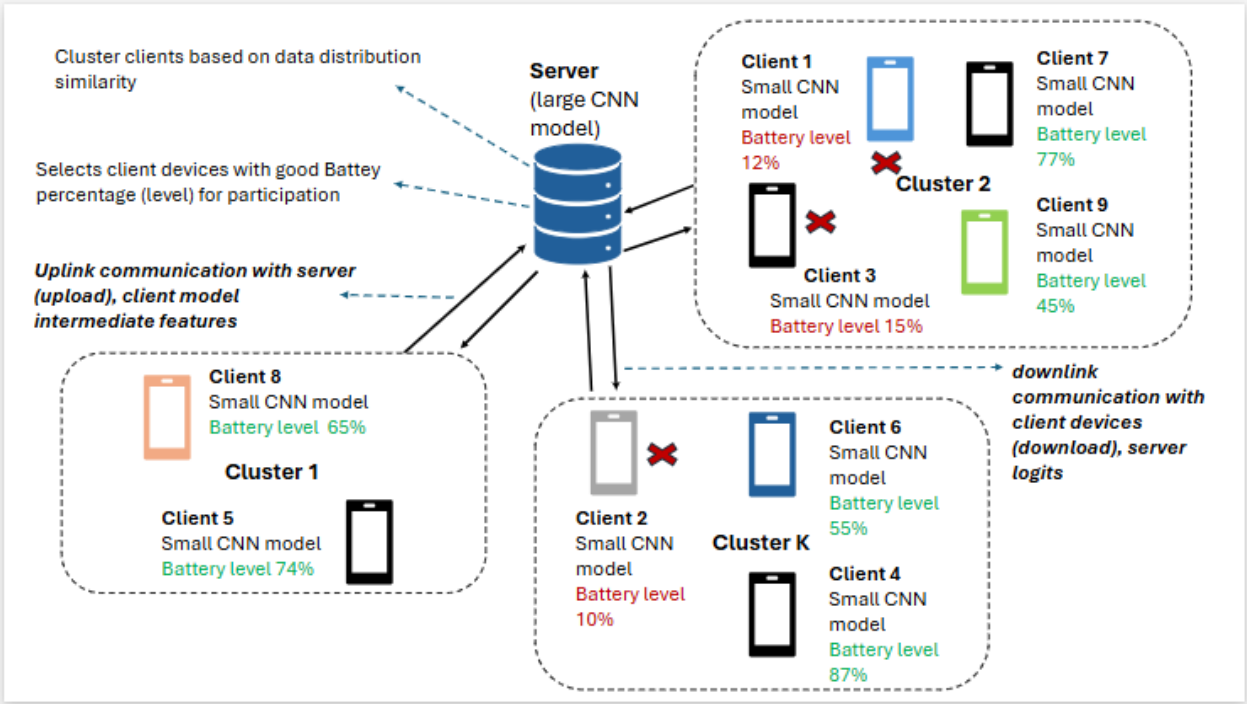


Figure 3.1: Architectural design of data distribution-based client clustering and battery aware client selection.

3.2.2 Client Data Distribution statistics

To simulate real world statistical data heterogeneity in federated learning, the chosen dataset will be partitioned in non-IID manner. Non-IID refers to data that is not independent and identically distributed, that is the data on each client device has varying class distribution such as, in MNIST a client might have one or two partition from two classes [78]. This implies the local data on each client device is not representative of the overall population distribution. This mimics real world scenarios where individual devices naturally accumulate data reflecting their specific usage patterns or geographical environment.

We adopt the Dirichlet distribution-based partitioning scheme, where the probability of data sample belonging to a client is drawn from a Dirichlet distribution. Dirichlet distribution is parametrized using a vector, called the concatenation parameters, Alpha [79]. This alpha parameter will control the degree or the severity of the non-IID ness across clients in the Dirichlet distribution. The degree of data imbalance corresponds to real-world deployments, where clients' local datasets differ in both size and composition. Lower values of alpha (e.g., 0.5) yield sharply skewed distributions, ideal for testing extreme distribution scenarios, while higher values (e.g., 1) produce milder, more balanced heterogeneity which is almost close to IID or heterogeneous data distribution among clients [80]. In a dataset containing classes and number of clients in the federated learning system, the Dirichlet distribution with parameter alpha governs the proportion of data from each class assigned to each client. Here on this study, we utilize the Dirichlet distribution to create non-IID data or heterogeneous data partitions among the number of clients.

3.2.3 Non-IID Client Data Distribution

Each client computes their local data distribution statistics, which involves mean and standard deviation calculation then sharing it with the central server [67]. The mean and standard deviation values might be calculated from the pixel vales of the local image data of a client. We assumed here that, as there is no raw data sharing which imposes minimal communication overhead. No raw data sharing is also good for the privacy concern, but there might still be an inference attack and applying other privacy model is beyond the scope of the study. Client data statistics is the foundation for the similarity-based client clustering which also enables informed client selection process. The mean and standard deviation of client dataset represented as:

$$Client\_statistics = \{(\mu_1, \delta_1), \ldots, (\mu_n, \delta_n)\} \tag{3.1}$$

Statistics computation for local data distribution, particularly involves [67] [81]:

Mean:

$$\mu_i = \frac{1}{n_i} \sum_{j=1}^{n_i} x_j \tag{3.2}$$

Standard deviation:

$$\delta_i = \sqrt{\frac{1}{n_i} \sum_{j=1}^{n_i} (x_j - \mu_i)^2} \tag{3.3}$$

Where, xi is individual sample from each client and ni is the number of samples on client i.

We are going to compute and share the local data distribution statistics of each client once at the beginning of the KD based FL process.

3.2.4 Client Energy Profile

The client devices are assigned with a device specific energy model, for the demand of power aware FL training and optimization of system efficiency over heterogeneous battery-powered edge devices. On our experiment, each client in the KD based FL system assigned a battery profile from a set of simulated values in the "energy profile" of the clients. The device profile is randomly assigned to each of the simulated clients in the KD based FL environment. Each client battery profile reflects the device category (high-end, medium-range and low end) in terms of their power consumption, as the power consumption for different end devices is different as well as their battery capacity. Client device category is modeled almost similar to [11] with typical battery capacity and power consumption models, but since the research was conducted bit earlier it does not consider recent advancements.

Here we have taken recent battery capacity and power consumption profiles from different battery powered device manufacturers verified sites as well as other performance ranking sites [82].

Low end devices are those devices which are less power efficient training a model, training model takes longer time and uses more energy per operation. They must run at high power levels for a long time to complete task, so the total energy used become high.

Those devices include, small android phones, basic IOT sensors with battery capacity ranging from 3000 – 4000 mAh and power consumption considering compute intensive tasks like model training range from 3-5 Watts. Mid-range devices have good balance of performance and efficiency that they can handle moderate training without excessive battery drain. They have more powerful and efficient that can compute tasks faster than low end devices. These mid-range decides include, old phones, commonly android phones, having battery capacity range 4500 – 5000mAh and with power consumption for compute intensive tasks from 4-7 Watts.

High end devices have lower power drain, that is more computation is done with fastest task completion which leads to lowest total energy consumption for a given computation. Overall, high end devices need more power but also more energy efficient than the others. Typical battery capacity of high-end devices ranges from 5000 – 6000+mAh with power consumption estimate of 5 – 14 watts.

The initial battery level of each client is set at random with percentage value of its total capacity. To ensure fairness and heterogeneity, the three categories of the devices are initialized with varying range of battery percentage. For using standardized energy measurement unit, the battery capacity is converted from milli amp hours to watt hours, it allows accurate comparison across all devices despite their voltage.

A battery recharge model is integrated to simulate real-world usage. When a client is not selected in the current round or in the coming consecutive rounds, its battery is recharged by a fixed recharge rate. This allows critically low-battery clients to recover over time and rejoin the training process as they might contain unique data for the training.

The energy consumption per round is calculated as the sum of computation and communication energy by also considering other background processes. Considering that most of the federated learning processes are carried out when the device is idle or connected to charger, the background processes are also related to this idle state of the device. The computational energy is computed from the product of computational time and power. The computation time is the time it takes to carry out the KD based FL training process and in the simulated environment measured in seconds, then converted to hours.

The power consumption that assumes compute intensive tasks like model training or inference, is different for the three device categories. The communication energy is also the product of communication time for sending an update between the server and the clients and communication power consumed. The communicational power consumed is categorized into two weather the client device is using a Wi-Fi/3G the consumption varies. So, the total energy consumed by client i in round t is then:

$$E_i(t) = E_{\text{computation}} + E_{\text{communication}} + E_{\text{background}} \tag{3.4}$$

Where, they represent the computation, communication and other background process energy consumption respectively.

This final computation value is deducted from the client initial battery to estimate the remaining battery. The battery levels of the clients will decrease as the simulation takes place and this battery profile of clients is assigned once for the whole communication round, but that does not include battery recharging state.

The communication energy usage depends on elapsed communication time and communication technology used. The Wi-Fi and 3G energy consumption is modeled as [83]:

Table 3.1: Communication energy consumption function with respect to elapsed time

	Download	**Upload**
Wi-Fi	$y = 18.09x + 0.17$	$y = 21.24x - 2.68$
3G	$y = 20.59x - 1.09$	$y = 15.31x + 2.67$

Where x is the communication duration in seconds with real world deviations accounted by including a scaling factor.

3.3 Proposed methodology

3.3.1 Similarity based client clustering

In practice, FL involves clients with diverse hardware, usage patterns, and data sources, leading to variation in both data sizes and distributions, known as data heterogeneity. Clustering clients with similar representations in federated learning address the issue of slow model convergence and degraded model performance [69].

Here our system clusters clients based on their data statistics similarity as reported in section (3.2.3). It ensures diversity and backup in case of client dropout, facilitating robust dropout mitigation. The clustering group clients based on their statistical similarity of their local data distributions. This aims to produce homogeneous client groups, which enables more effective knowledge transfer with improved learning process.

K-means clustering groups clients based on their data distribution statistics-specifically the mean and standard deviation of data points using Euclidean distance to measure similarity. Clients with in each cluster exhibit grater intra-cluster similarity than inter cluster similarity. We selected K-means as it effectively clusters clients based on our data distribution and suited to our KD based FL scenario. The optimal number of clusters K is determined using silhouette coefficient and Davis Bouldin index, which the server employs to asses clustering quality, in our scenario we also assigned tested the cluster value rather than measured ones by these mechanisms.

3.3.2 Battery-aware client selection

Device dropout from energy depletion poses a major challenge in FL. to mitigate this, we integrate battery level aware client selection in to each communication round, prioritizing clinerts based on their current energy levels models via realistic simulated battery dynamics.

In every round, the server evaluated battery levels using device specific profiles from three categories (Section 3.2.4), where initial levels are randomized with in each category's minimum capacity. Clients meeting a predefined battery threshold are eligible; those below it are excluded to prevent stability and convergence speed.

Available battery energy is computed as:

$$Battery = \text{battery capacity} \times (\text{battery percent}/100) \tag{3.5}$$

Were, the battery capacity implies capacity for those three different categories, and the battery percent is assigned randomly to each client.

To preserve fairness and heterogeneity of devices, clients are selected from each cluster using a weighted random selection scheme. This ensures that not only high-end devices are consistently chosen, but also mid-range and especially low-end devices get the opportunities to participate in training, while still giving priority to more energy-capable clients.

This mechanism provides a balance between energy efficiency and fair client participation, eventually reducing dropout rates and enhancing the robustness of the federated learning process.

3.3.3 Clustering based Battery-aware Client selection

Our research strategy client clustering based battery aware clients selection enhances robustness and performance in KD based FL systems by minimizing dropout impacts and ensuring fairness among various device resource ranges.

Data similarity clustering first groups clients with comparable distributions such as mean, standard deviation, skewness fostering homogeneous subgroups among the clusters. With in each cluster, battery aware selection prioritizes energy stable clients above a threshold, reducing mid round dropouts and disruptions.

For the client dropout mitigation, clustered peers provide substitute updates, preserving the dropped client's data characteristics during global aggregation. Selected clients then perform feature based KD-FL training, exchanging features and logits with the server for efficient knowledge transfer and aggregation.

3.3.4 Knowledge distillation based federated learning framework

The KD based FL framework follows the approach in [14], and incorporates split learning to improve communication and computational efficiency on resource-constrained devices. In this architecture, the global model is vertically partitioned, with the initial layers assigned to the client side model and the subsequent layers to the server side model. Computationally intensive components, such as the classification head, are hosted on the server to leverage its greater processing power. conversely, light weight feature extracting layers reside on client devices, allowing local computation without excessive resource demands.

The selected clients download the current client-side portion of the global model. They then perform a forward pass on their local datasets, training the client-side model. Instead of sending gradients or full model parameters, clients extract and output intermediate features or logits. Then on the current communication round, the clients will send these features/logits to the server which significantly reduces communication overhead compared to traditional FL, as features/logits are smaller than model parameters/gradients. The server after receiving features/logits from multiple clients, their update will be aggregated for the server-side model. The server then performs the remaining computation of classification or other tasks, and the server-side model parameters will be updated based on aggregated features for the coming rounds. In our KD based FL set up, distilled knowledge (features/logits) propagated in both directions of client to server and vice versa. This "feature-based KD" significantly minimizes communication as compact knowledge representations are sent back to clients without full model weights.

The clients apply two sources of knowledge during their local training; they use their own hard labels from their private local datasets and soft labels/logits propagated from the server's global model. Since the knowledge from the client data is very small, it is not enough for making generalized model.

3.3.5 The training process

Our overall KD based FL training process coordinating the interaction between the server and the clients which also involves client clustering based battery aware client selection over multiple communication rounds:

Initialization: the global model is randomly initialized on the server and, split into client side and server-side partitions. The participating clients assigned unique non-IID dataset partition and a battery profile with initial battery level. In federated learning system, originally the training will take place on client devices that are idle (e.g., mobile device that is not being used by the user) or in changing state.

Client data distribution statistics sharing: all the available clients' compute their local data distribution statistics, mean and standard deviation of their local data points and then transmitting those to the central server.

Client clustering: the server upon receiving client statistics, it performs k-means clustering that groups clients with similar data distribution separately. This cluster will occur once when the communication round between the clients and the server starts.

Per-round client selection: for each communication round, the server selects subset of clients to participate, by filtering and discarding clients whose battery level below the critical threshold. The battery-aware client selection process engage weighted random selection, from each cluster prioritizing clients with higher battery levels and also low end devices still also has a chance of being selected to ensure not to lose unique data.

Battery recharge: some of the unselected clients in the current round or in the subsequent rounds, their battery is recharged by a fixed recharge rate. In this way critical battery clients may get the chance to rejoin the selection process.

Local training on selected clients: each selected client using client-side model performs local training on private non-IID dataset. By implementing forward pass and backward pass using client-side model. This training incorporates both the clients' local hard labels and distilled knowledge received from the server.

Feature/logit upload to server: after local training performed by the selected clients, the clients computed intermediate features/logits is sent to the server.

Server training with KD: the server aggregates the received features/logits from all participating clients, then feeding the aggregated features into the server-side model for computing and updating the model parameters. Then, server based on its updated global knowledge, it generates distilled knowledge (logits) to be broadcasted back to clients in subsequent rounds.

Dropout mitigation: if a selected client drops out during communication round e.g., due to network issue, system failure, its contribution to the current round will be lost. However, the data similarity-based clients clustering can provide compensatory update to the server before aggregation in replacement to the unintentionally mid training dropped out client. This ensures the statistical representation of the dropped-out clients' data is still partially included in the global aggregation, reducing the impact of dropout on model performance.

Battery level update: in the simulated environment, at the end of each round, battery levels of participated and non-dropped clients are updated by deducting energy consumed during their training participation and other background processes (in an idle state).

The server using the updates from participating clients to produce an updated global model. This training steps repeat in the subsequent communication rounds until convergence criteria like, peak maximum global accuracy, required number of rounds has reached.

3.4 Evaluation metrics

A comprehensive set of evaluation metrics is employed to assess the effectiveness of the proposed dropout-resilient, battery aware KD based FL framework. Beyond conventional accuracy, the evaluation emphasizes robustness to client dropout, convergence behavior, and resource efficiency, which are central to the objectives of this study.

Global model quality is assessed using classification accuracy on held out test dataset to measure generalization, along with dropout resilience, quantified by performance degradation under varying dropout rates. Convergence speed is evaluated by analyzing the number of communication rounds required to reach a stable accuracy threshold. In addition, energy is measured through per round client participation and energy consumption trends, reflecting the impact of the proposed battery aware client selection mechanism. collectively, these metrics provide a comprehensive evaluation of the system's ability to maintain stable and reliable performance in presence of device dropout.

3.4.1 Model performance

Global model accuracy: the classification accuracy of the aggregated global model on a held-out test dataset, that represent the overall data distribution at the end of each communication round. This is the key measurement of the model's learning effectiveness.

Global model loss: the MSE loss of the aggregated global model on the held-out test dataset, indicating the model's prediction error and convergence behavior.

3.4.2 System robustness

Energy consumption per client in each round measures the average energy consumed by selected and actively participating clients in each communication round, reflecting the resource demands of the proposed KD based FL process. comparing energy consumption under battery aware client selection with random client selection highlights the effectiveness of the proposed approach in selecting more sustainable participants.

Remaining battery level of active clients at the end of each communication round is tracked to evaluate the system's ability to manage client energy and sustain participation over time. Higher remaining battery levels indicate improved robustness and more efficient supervision of client energy resources, contributing to reduced dropout and stable training.

3.5 Implementation and development tools

This section outlines the details of the software, hardware, and programming environments used for the implementation, simulation, and evaluation of our proposed knowledge distillation based federated learning system. These tools were selected to fulfill the need for robust deep learning capabilities, efficient simulation of distributed FL environment, and for the reproducibility of results. The following subsections provide an outline of the development environment.

The vast collection of scientific libraries was ideal for this research. to its rich selection of scientific libraries.

3.5.1 Software environment

Programming language:

Python is the primary programming language that is used, and it is selected due to its simplicity, wide support for machine learning research, and compatibility with deep learning frameworks. The broad and rich collection of scientific libraries was ideal for this research.

Deep learning framework:

PyTorch was utilized as deep learning framework due to its flexibility for defining neural network architectures, and widespread adoption for research. It enables implementation of complex architectures like federated learning, split learning, and knowledge distillation.

Operating systems: Ubuntu 22.04.5 LTS was used for local experiments. NVIDIA DGX server Ubuntu 20.04.6 LTS (GNU/Linux 5.4.0-204-generic x86) used for its stability and wide support for GPU acceleration in machine learning research for training complex deep learning models. This server contains 8 high performance GPUs, each processing 128GB of memory. Windows 11 was also used as an operating system on our local environment for development, this provides a flexible environment for testing, evaluating and perform other development functions.

Google Colab: served as a cloud-based platform for testing and rapid prototyping, providing free version GPU access for small-scale experiments. It offers a convenient platform for code testing and demonstrating.

Development and simulation environment: our experimental setup used both dedicated primary server infrastructure and google Colab as secondary cloud-based platforms to manage computational demand and scalability.

3.5.2 Hardware environment

The experimental simulations were conducted using high performance computing resources to support the computational demands of federated learning experiments.

Primary server: an NVIDIA DGX Server was used as the primary computing platform. The system is equipped with 8 x NVIDIA GPUs with a total of 128 GB GPU memory (16 GB per GPU), a multi-core processor optimized for parallel computation, and 512 GB system memory (RAM).

Cloud environment: Google colab was employed as a secondary computing platform, providning access to NUVIA Tesla GPUs, such as T4 model, with 16 GB of GPU memory.

Chapter 4

Result and Discussion

This section delivers a comprehensive and reproduceable description of our experimenting environment, dataset and model configurations.

4.1 Dataset

Our experimental evaluation employs the MNIST dataset, which is widely adopted in federated learning due to its widespread use and simplicity, it is also a benchmark dataset for image classification. MNIST dataset contain 70,000 grayscale images each of size 28x28 pixels of handwritten digits (0-9), split into 60,000 training images and 10,000 test images. MNIST characteristics makes it ideal for validating federated learning approaches, particularly those line non-IID data distribution challenges. Relatively, it has small size that allows for rapid iteration and experimentation, delivering straight forward metric for performance evaluation.

The primary task is classification on MNIST dataset that involves correctly labeling each image to its corresponding digit class, performing multi-class classification, and where the global federated model learns to classify digits. In addition, generating non-IID variants of MNIST data is performed, which is crucial experimental task that introduces real-world challenge of data heterogeneity among client devices. This is essential for robustness and practical applicability of our proposed system.

4.1.1 Data partitioning

To reflect real-world FL environments, in which clients exhibit statistical heterogeneity, data is partitioned into non-IID data partitions using the Dirichlet distribution approach.

This method allows for controlled generation of non-IID data by assigning a proportion of each class to clients based on a Dirichlet parameter "alpha". The data partitioning and sampling process ensures that data is not uniformly distributed across classed for each device, creating the desired heterogeneity. Here in our simulation environment, we have used alpha value 0.5. A detailed description of the data partitioning and the mathematical formulation of Dirichlet allocation is provided in the Appendix A.1.

Table 4.1 shows how data is being partitioning in our simulation among 40 total number of clients via non-IID Dirichlet distribution.

multirow

Table 4.1: Dataset Partitioning Setting

Dataset	**No. of classes**	**No. of partitions**	**Partitioning method**	**Partitioning setting**
MNIST	10	40	Non-IID Dirichlet distributed	Distribution parameter ($\alpha = 0.5$)

4.1.2 K-means client clustering by Client data distribution

The clients perform calculation of statistics (mean and standard deviation) of the pixel values from the flattened image data samples. After the server received each client statistics, a vector of the statistics is constructed, then serve as input to the K-means clustering algorithm.

The optimal number of clusters K in the K-means clustering, is typically found by using metrics namely silhouette coefficient and Davis Bouldin index. The sever employ those metrics to evaluate the quality of clustering and determining the appropriate K, much detail is included in Appendix A.3. clients within the same cluster possess statistically similar local data distributions based on the proximity of their statistical vector value. This clustering minimizes intra-cluster variance and ensures that if dropout occurs, clients within the same cluster provide as replacements. An appendix showing details of the similarity calculation and clustering algorithm configurations are provided in the Appendix A.2.

4.2 Model Architecture

A simple CNN architecture utilized suitable for the MNIST classification is employed, which is split between client and server in our knowledge distillation based federated learning setup. The clients handle the feature extraction while the server performs the heavy classification task, enabling resource-efficient training on low powered clients.

The client-side model (feature extractor) runs on resource constrained edge devices. It contains a sequential convolutional block designed for initial feature extraction. Three convolutional layers, that processes the single channel 28x28 input image to produce 32 feature maps, that applies ReLU activation. Two Max-pooling layers for dimensionality reduction, that reduces dimension by half twice such as 28x28 to 14x14. Finally, there is feature vector extraction of 128x7x7 dimensional output.

The server-side model (classifier) resides at the central server which acts as the global model and receives the extracted features from the clients. It consists of a sequential block of fully connected layers with dropout and normalization. Dense layers that receive flattened features of 128x7x7 dimension and transforming them to 512 features. Then the final layer outputs 10 logits, corresponding to the 10 classes of MNIST digits. The split learning implementation involves feature based knowledge transfer, in which the client model extracts features that when flattened serve as input for the server model. The client model output dimension matching with the server input dimension enables the split execution, where clients carry out the initial feature extraction and the server completes the classification task. The detail of model structures is shown on Table 4.2

4.2.1 Model parameter

Here is the hyper parameter values used in the CNN models on the MNIST dataset, for both for the client side and server-side model. The parameters demonstrate that our models effectively learned from the data while maintaining efficient training time, additionally they are crucial for reproducibility. on the below Table 4.3, it shows the details of hyper parameter that we have used for training the model both at the clients and the server.

Table 4.2: Model structure for MNIST

Property	**Student model (client model)**	**Teacher model (server model)**	**Description**
Input layer	28 × 28 gray scale image	Extracted features from client model (128 × 7 × 7 dimension)	The data fed in to the models.
Convolutional layers	Conv2D: 32×28×28 (output shape) Conv2D: 64×28×28 (output shape) Conv2D: 128×14×14 (output shape), by Maxpool reduced by half to 128×7×7. Final classification layer = 10 (dimension)	None	Layers responsible for extracting features from image data
Final layer	Linear layer (128×7×7)	Linear: 1^{st} fully connected layer, 128×7×7 (input dimension) Linear: 2^{nd} fully connected layer Linear: 3^{rd} fully connected layer, 10 (output dimension)	Last layer responsible for the classification logits (10 classes)

Table 4.3: The parameters used on our experiments with their respective assigned values

Hyper parameter	**description**	**value**
Client optimizer	Stochastic Gradient Descent (SGD)	–
Client Learning Rate	Step size for SGD	0.01
Client Momentum	Momentum for SGD	0.9
Client Weight Decay	L2 regularization for SGD	5e-4
Client Epochs	number of local training epochs on each client per communication round	7
Server Optimizer	Adam optimizer	–
Server Learning Rate	Step size for Adam	0.001
Server Epochs	number of epochs for server-side model training	5
Batch size	number of samples processed in one forward/backward pass	32
Loss Function	standard cross entropy loss for local client training cross entropy loss for evaluation purposes KL divergence loss for knowledge distillation	KL loss

4.3 Knowledge Distillation parameters

Feature-based KD was integrated to enhance the generalization of the federated learning system. Clients learn both from their own labels and the server's softened logits, improving performance despite non-IID local datasets. This facilitates communication-efficient model training, enabling clients with smaller models to learn from the global server model's richer representations.

The critical hyperparameter governing our Knowledge Distillation framework is Temperature (T). T is applied to the logits of both the teacher and student networks before the Soft Max function is computed. Temperature is used to control the smoothness or softness of a model's output probability distribution. The standard Soft Max function converts a model's raw output scores (logits) into a probability distribution. A higher T value "softens" the probability distribution, making the probabilities of all output's classes more uniform and less peaked. A low temperature results in a "harder" or more peaked probability distribution, similar to standard Soft Max output. Here we have used value of 2.0 for the T, we have chosen a lower temperature value because we want the clients to learn more from their local dataset and less from the teacher's "softened" outputs for some information for the clients to learn from.

4.4 Heterogeneous Energy Profile for Battery Aware KD based FL

In our KD-based Federated Learning system, 40 simulated clients are established and from those a target number of 20 clients are selected for participation in each training round. The selection is performed from each of the 4 cluster that ensure data diversity, with battery aware client selection, in which clients with higher battery level have higher selection probability but low battery clients retain a non-zero chance to participate in order to maintain data diversity and fairness. Clients below a critical battery threshold are excluded from participation.

Each simulated client is assigned a battery or energy profile corresponding to high, medium and low – end devices. The battery modeling for each client device, which tracks their battery profiles and consumption during the training process. Battery capacity is modeled in Watt-hours (Wh), with computation and communication consumption deducted in each communication round.

In FL, it is considered that the devices are idle or not in use by the user during training process, allowing for focused analysis of energy consumption due to KD-FL task. The battery model accounts for, energy consumption due to computation of the client devices training consumption, communication between the client devices and the server both for upload and download communication energy consumption and Idle or other background processes of the device consumption.

The computational energy consumed by each client is computed as product of computational time and power draw assigned by each client. This computation is different for low-end, mid-range and high-end devices as stated in the Table 4.4. The communication energy consumption is calculated as a product of communication time (both for the upload of feature to the server and download of logits to the clients) and communication power (could be Wi-Fi or 3G based on what the device is using)

Table 4.4: Classification of device ranges based on their battery capacity and power consumption profiles

Device	**Typical battery capacity per milliampere-hour**	**Power consumption per watt**	**Processing power profile**
Low-end	3000 – 4000 mAh	3 – 5 watts	These devices are power inefficient, so they must run at high power levels (energy per operation) for a long time to complete a task, leading to high total energy used and computation time.
Medium-range	4500 – 5000 mAh	4 – 7 watts	They have good balance of performance and efficiency and can handle moderate training tasks without excessive drain or heat.
High-end	5000 – 6000+ mAh	5 – 14 watts	They have high performance, high efficiency as they are built on latest processors. They complete tasks fastest, leading to lowest total energy consumption for a given task, being more energy efficient.

Table 4.4 we have categorized client devices into three groups based on their hardware capabilities and power consumption profiles. The grouping provides a representative model for heterogeneous client populations, enabling more accurate evaluation of our federated learning approach under real-world conditions.

4.5 Baseline and experimental variants

Three main experimental conditions are compared here namely the baseline (full client participation without dropout), client dropout without replacement, and client dropout with replacement (our system). The baseline is our main comparison point; it is client clustering based KD based FL system. It is an ideal condition in which all selected 20 clients complete training without dropout from the training process. In our scenario, this represents the upper bound of performance for our system without considering dropout.

We introduced a mechanism for simulating client dropout, in which selected clients randomly dropout at some ratio such as 40%, 60 % and 80% of the simulated 20 clients will dropout from the training process. They are not replaced reducing effective training participation. We can control the dynamic dropout rate of clients, as the client dropping increases clients have reduced opportunities to join the training rounds, leading to performance degradation.

In our system, dropped clients will be replaced by others from same cluster; if none are available replacements are drawn from other clusters. This highlights our key mitigation strategy, and also maintains cluster representativeness. The battery-aware client selection is involved in all three simulation categories, by ensuring a fair comparison under conditions where battery life is considered for initial selection. But, even with battery-aware selection, clients might still dropout due to other real-world issues like network connection instability and others. Therefore, the dropout simulation and replacement algorithms are crucial for demonstrating robustness of the system in realistic and imperfect environments.

For each of the above three scenarios, the performance under MNIST will be plotted and compared against each other, to see the effectiveness of our replacement algorithm.

4.6 Experimental results

To evaluate the convergence performance of our system under clustered client settings with varying client dropout percentages. In federated learning, convergence refers to the stabilization of the global model's performance across communication rounds, where client updates become aligned and further training yields minimal improvements.

In the setup, the participating 20 clients were grouped in to 4 clusters, where clients with in the same cluster share similar data characteristics, that clustering allows dropped clients to be replaced with others from same group. For comparison, we evaluated our clustered client selection and replacement across different dropout rates and full participation analyzing accuracy and loss curves.

The table 4.5 below shows the KD based FL parameter for the global model training.

Table 4.5: Experimental Parameters

Description	**Value**
Total clients	40
Clients per round	20
Number of rounds	70
Number of local epochs	1
Number of server epochs	5
Batch size	64
Momentum for SGD	0.9
KD parameter (T)	0.2

4.6.1 Experiment 1

On the first experiment, there are plots for test accuracy and training loss against communication rounds. These visualizations serve to validate the system's performance, stability, and its effectiveness in real-world complexities. We also evaluate how client dropout and replacement strategy affect global model convergence. The accuracy versus communication rounds illustrates, final global model performance under different client device dropout settings showing whether the proposed system can maintain accuracy close to the full participation of all the 20 clients.

This direct comparison provides strong evidence for the benefits our proposed method. The training loss versus communication rounds provides complementary evidence of training stability, showing whether the system converges smoothly or suffers from oscillations due to missing client updates. Stable curves in general indicate well-tuned hyperparameter and robust aggregation mechanism of the knowledge distillation based federated learning system. All these results demonstrate that the proposed clustering-based replacement and battery-aware client selection mechanisms mitigate the negative effects of client dropout, ensuring faster convergence, higher accuracy, and more stable training as compared to random dropout scenarios without replacement.

Accuracy performance over rounds

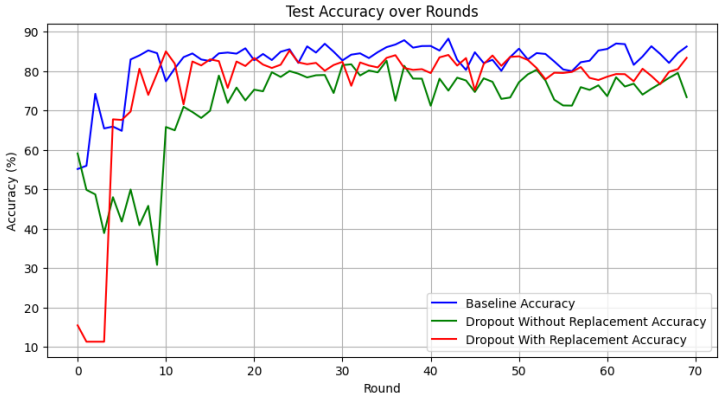
Here the global model (server model) test accuracy over the given 70 communication rounds is tracked.

During the communication rounds the relevant hyper parameters that control the knowledge distillation based federated learning system dynamics, those hyperparameter are such as in our scenario 20 clients selected in each communication round, non-IID data partitioning, clustering parameters on the performance.

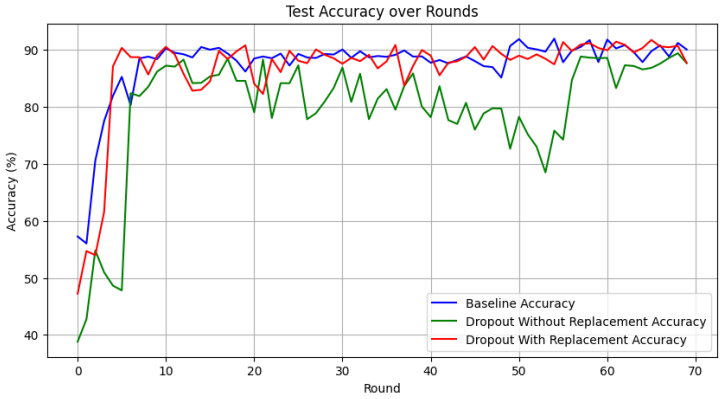
We will present combined accuracy over communication rounds graphs for different dropout scenarios to facilitate comparison. The clients are intentionally dropped out randomly in order to see the effect when different amount of clients' dropout from the training process.

At 20 percent, 50 percent ,80 percent client dropout, there is a single combined graph shown as below plots in which inside each of the graphs it shows (1) full participation (no dropout) of the selected 20 clients plot, (2) 40%, 60% and,80% from the selected clients randomly dropped out plot and, (3) our system's performance in which 40%, 60% and,80% of the dropped clients replaced by clients from other unselected simulated clients from same cluster. We give more attention to substitution from the same cluster for dropped out clients in that it avoids stale or biased updates common in dropout scenarios.

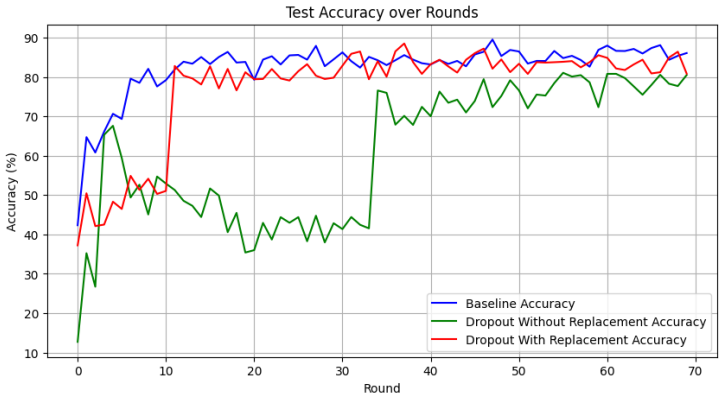
On Figure 4.1, the global model test accuracy versus communication round graphs illustrates that cluster-based client replacement for the dropped-out mechanism has mitigated performance degradation caused by client dropout. Replacing clients ensures that global model continues to receive an update from similar data distributions of those who had dropped out from the training that improves convergence speed and global model accuracy.



(a) Global model test accuracy for 40% client dropout.



(b) Global model test accuracy for 60% client dropout



(c) Global model test accuracy for 80% client dropout

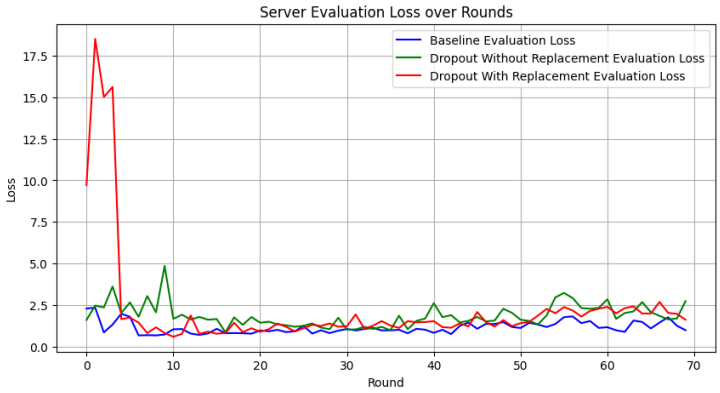
Figure 4.1: Global model test accuracy over communication rounds under different client dropout levels (40%, 60% and 80%) on MNIST dataset.

The baseline accuracy (blue curve), archives comparatively higher and most stable performance which is the starting point for comparison. It reflects full client participation without any dropout and disruption. The green curve shows a client being dropped out without replacement, and noticeable degradation in model performance is observed. The global model accuracy converges more slowly due to reduced data diversity and representation caused by sudden absence of several clients, leading to less informative updates being aggregated by the server. It can be seen that the drop in accuracy becomes more severe as dropout rate increases, signifying system's vulnerability to large scale client unavailability. The other curve depicts that clients that has dropped out being replaced by other clients from same cluster (red curve). It exhibits marked improvement compared to the dropout only step, that ensures each cluster remains represented even after dropout. As a result, the replacement algorithm maintains accuracy much closer to the baseline particularly at lower dropout rate, showing faster recovery and more stable convergence compared to the non-replaced client dropout scenario.

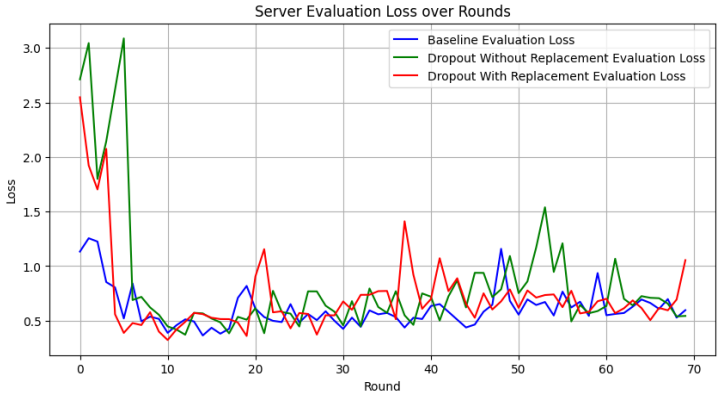
When the dropout rate becomes extreme performance still falls short of baseline indicating, while replacement alleviates the problem it might not compensate for large scale client availability as it might lose representative client.

Loss curves

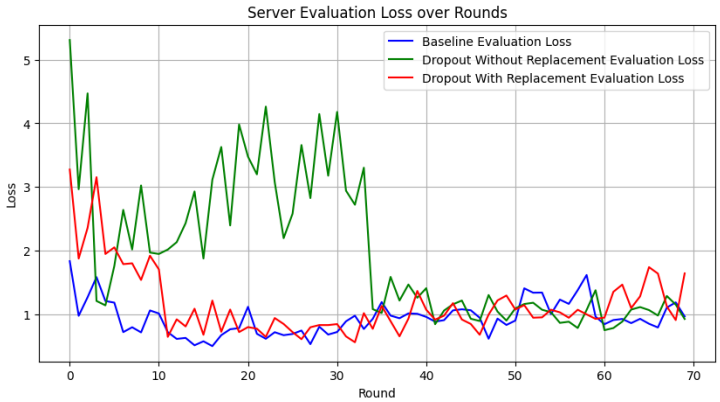
Alongside each of the previous combined accuracy graphs, here Figure 4.2 provide corresponding loss versus communication rounds curves, offering a comprehensive view of the training dynamics and convergence behavior.



(a) Global model loss for 40% client dropout.



(b) Global model loss for 60% client dropout



(c) Global model loss for 80% client dropout

Figure 4.2: Global model loss over communication rounds under different client dropout levels (40%, 60% and 80%) on MNIST dataset for the respective accuracy on the above figure.

4.6.2 Experiment 2

On this experiment, the battery dynamics across training rounds is crucial for accurately simulating real-world federated learning scenarios, especially when resource-constrained edge devices such as smart phones and IoT divides are targeted. Device batteries represent a critical limiting resource impacting client participation and availability during training. Here on our system, we model device specific battery profiles and simulating energy consumption during computation and communication activities, these experiments capture the real-world behavior of client devices. The client devices are also categorized into three tiers namely, high end, mid-range and, low end devices based on their battery capacity and the processing power of the device. The client selection strategy prioritizes devices based on current battery levels, ensuring clients with sufficient energy from different category of devices participate in that current round by reducing mid-training dropout rates and preserving data diversity. The selection of clients for participation make sure that it's not only favoring high end devices.

Battery-aware KD based FL training

The Bar chart on below Figure 4.3 demonstrates the battery depletion of sample clients that are selected using a standard random selection in the KD based FL framework. As shown, random selection approach fails to account for diversified battery capacities across heterogeneous device pool. As a result, clients experience battery depletion and dropout from the training process. The figure clearly establishes the necessity for resource aware client selection from all device categories.

The other bar chart on Figure 4.4 illustrate what is happening in a single communication round, it visualizes the battery status of the participating clients in the battery in watt-hours versus participating clients plot. The initial battery, the calculated consumed battery (which is different for different category) and, the final battery value after the consumption deducted from the initial battery for the participating devices in a particular selected single communication round. as clients are selected based on their battery level they have good battery to participate in the current training round without dropping out due to battery depletion issue.

Figure 4.3: Random selection of clients and battery consumption illustrated resulting client dropout due to battery draining in a single communication round.

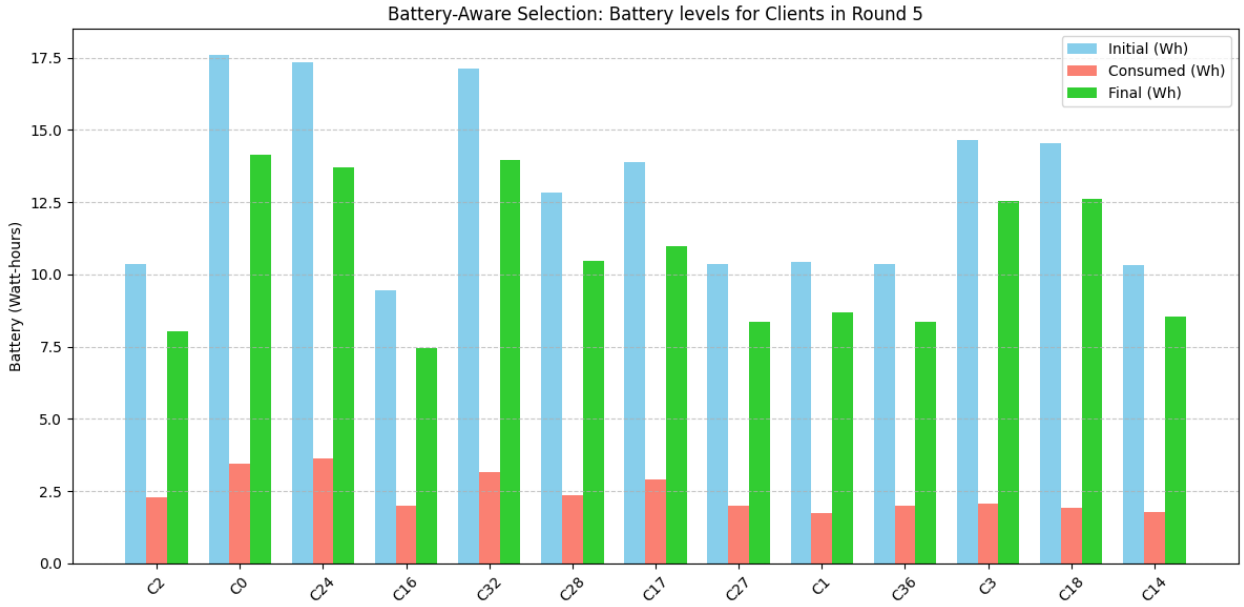


Figure 4.4: A comparative view of battery consumption across Low-end, Medium-range, and High-end device profiles for the battery aware client selection in a single communication round.

4.7 Discussion

The clustering mechanism could sample diverse data from all statistical groups (Low-end, Mid-range, High-end) ensuring global model receives more representative update in each round. The client clustering based on their data distribution similarity successfully create redundancy mechanism that minimize the negative effect of device dropout. It prioritizes clients based on their remaining battery levels from all categories, ensuring equitable client participation across the federation. It will reduce the frequency of premature client dropout due to battery depletion and enhance stability of KD based FL by improving overall convergence of FL training process.

***RQ*1** What traits of client devices can be used to design a client selection mechanism to mitigate the impact of device dropout to retain an acceptable performance of the KD based FL systems?

Integrating battery level awareness and cluster-based client replacement, our system ensures that when clients with network and other issues withdraw, an alternative client with similar data characteristics and sufficient energy resources are selected to continue participation. It preserves statistical representativeness and consistent participation across clusters throughout the training. Consequently, the approach reduces impact of client dropout in KD based FL training process.

Our system utilizes a collaborative mechanism of client selection by observing both the data distribution similarity for clustering and battery-aware selection by prioritizing good battery clients. This method aligns with prior finding of Heqiang Wang et al. (2023) [18], which also alleviates client dropout issue, unlike this work, our system converges slightly faster as well as they use random clustering of clients during training. Our system also implements resource aware client selection. As of this study conducted, there was no previous work that has been done regarding client dropout in KD based FL system.

This research signifies an applicable path for deploying KD based FL in non-IID and unstable environment such as mobile devices or IoT networks where battery and other resource constraints are major concerns. As a result, this makes FL more practical for edge computing. Despite a better global model performance achievement, the system has some limitation in dynamically adapting the clustering and selection mechanism to rapidly changing device statuses and ensuring scalability.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

This study examined the impact of device dropout on the performance of knowledge Distillation based Federated Learning systems operating in heterogeneous and resource constrained small edge devices. Device energy limitations and unreliable participation were the major factors that affect the stability and performance of the global model. To address these challenges, an energy aware Knowledge Distillation based Federated Learning framework incorporating data distribution similarity-based client clustering was proposed. The objective of the framework is to reduce client dropout by prioritizing clients with sufficient battery levels while improving knowledge transfer consistency through data similarity-based grouping. Experimental results show that the proposed approach effectively reduces client dropout and improves global model accuracy and its convergence stability towards the optimum as compared to conventional Knowledge Distillation based Federated Learning methods. These findings confirm that interacting data similarity and energy awareness enhances the robustness and practicality of Knowledge Distillation based Federated learning systems for real world deployment.

5.2 Future Work

Future research may investigate several findings to further enhance dropout resilience in federated learning. A dynamic clustering algorithm that can adapt to scalability and real-time data distribution and device heterogeneity by enhancing selection flexibility. It can also investigate an alternative Knowledge distillation mechanism such as a strategy that employ multiple teacher model with one student model and other knowledge transfer mechanism. A system that ensures the client selection techniques uphold privacy and security standards against malicious clients or adversarial attacks. Investigating the compatibility of our strategy with other federated learning enhancement like differential privacy or personalization.

References

[1] Chuhan Wu, Fangzhao Wu, Lingjuan Lyu, Yongfeng Huang, and Xing Xie. Communication-efficient federated learning via knowledge distillation. *Nature communications*, 13(1):2032, 2022.

[2] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

[3] Alessio Mora, Irene Tenison, Paolo Bellavista, and Irina Rish. Knowledge distillation for federated learning: a practical guide. *arXiv preprint arXiv:2211.04742*, 2022.

[4] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and trends® in machine learning*, 14(1–2):1–210, 2021.

[5] Alexander Brecko, Erik Kajati, Jiri Koziorek, and Iveta Zolotova. Federated learning for edge computing: A survey. *Applied Sciences*, 12(18):9124, 2022.

[6] Yuchang Sun, Yuyi Mao, and Jun Zhang. Mimic: Combating client dropouts in federated learning by mimicking central updates. *IEEE Transactions on Mobile Computing*, 23(7):7572–7584, 2023.

[7] Laiqiao Qin, Tianqing Zhu, Wanlei Zhou, and Philip S Yu. Knowledge distillation in federated learning: A survey on long lasting challenges and new solutions. *arXiv preprint arXiv:2406.10861*, 2024.

[8] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine*, 37(3):50–60, 2020.

[9] Eunjeong Jeong and Nikolaos Pappas. Battery-aware cyclic scheduling in energy-harvesting federated learning. *arXiv preprint arXiv:2504.12181*, 2025.

[10] Filipe Maciel, Allan M de Souza, Luiz F Bittencourt, Leandro A Villas, and Torsten Braun. Federated learning energy saving through client selection. *Pervasive and Mobile Computing*, 103:101948, 2024.

[11] Amna Arouj and Ahmed M Abdelmoniem. Towards energy-aware federated learning on battery-powered clients. In *Proceedings of the 1st ACM workshop on data privacy and federated learning technologies for mobile edge network*, pages 7–12, 2022.

[12] Maoxuan Yan, Qingcai Luo, Bo Zhang, and Shanbao Sun. Solving client dropout in federated learning via client similarity discovery and gradient supplementation mechanism. In *International Conference on Algorithms and Architectures for Parallel Processing*, pages 446–457. Springer, 2023.

[13] Dekai Zhang, Matthew Williams, and Francesca Toni. Clustered federated learning via embedding distributions. *arXiv preprint arXiv:2506.07769*, 2025.

[14] Chaoyang He, Murali Annavaram, and Salman Avestimehr. Group knowledge transfer: Federated learning of large cnns at the edge. *Advances in neural information processing systems*, 33:14068–14080, 2020.

[15] Bingyan Liu, Nuoyan Lv, Yuanchun Guo, and Yawen Li. Recent advances on federated learning: A systematic survey. *Neurocomputing*, 597:128019, 2024.

[16] Zhiyuan Wu, Sheng Sun, Yuwei Wang, Min Liu, Xuefeng Jiang, Runhan Li, and Bo Gao. Knowledge distillation in federated edge learning: A survey. *arXiv preprint arXiv:2301.05849*, 2023.

[17] Jiawei Shao, Yuchang Sun, Songze Li, and Jun Zhang. Dres-fl: Dropout-resilient secure federated learning for non-iid clients via secret data sharing. *Advances in Neural Information Processing Systems*, 35:10533–10545, 2022.

[18] Heqiang Wang and Jie Xu. Combating client dropout in federated learning via friend model substitution. *arXiv preprint arXiv:2205.13222*, 2022.

[19] Yujing Chen, Yue Ning, Martin Slawski, and Huzefa Rangwala. Asynchronous online federated learning for edge devices with non-iid data. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 15–24. IEEE, 2020.

[20] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[21] Yu Qiao, Chaoning Zhang, Huy Q Le, Avi Deb Raha, Apurba Adhikary, and Choong Seon Hong. Knowledge distillation in federated learning: Where and how to distill? In *2023 24st Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pages 18–23. IEEE, 2023.

[22] Hassan Salman, Chamseddine Zaki, Nour Charara, Sonia Guehis, Jean-François Pradat-Peyre, and Abbass Nasser. Knowledge distillation in federated learning: a comprehensive survey. *Discover Computing*, 28(1):145, 2025.

[23] Neptune.ai. Knowledge distillation: Principles, algorithms, applications. `https://neptune.ai/blog/knowledge-distillation`, 2024. Accessed: September 22, 2025.

[24] Jianfei Zhang and Yongqiang Shi. A personalized federated learning method based on clustering and knowledge distillation. *Electronics*, 13(5):857, 2024.

[25] Liu Ying, Yu Lu, Guoping Zhu, Shuang Liu, Ruizhi Li, Ke Yang, and Jingmei Wang. Blockchain based semi-decentralized personalized federated learning. *Available at SSRN 5344325*.

[26] Wanli Ni, Huiqing Ao, Hui Tian, Yonina C. Eldar, and Dusit Niyato. Fedsl: Federated split learning for collaborative healthcare analytics on resource-constrained wearable iomt devices. *IEEE Internet of Things Journal*, 11(10):18934–18935, 2024.

[27] IBM Research. Knowledge distillation.

[28] Jinghui Zhang, Jiawei Wang, Yaning Li, Fa Xin, Fang Dong, Junzhou Luo, and Zhihua Wu. Addressing heterogeneity in federated learning with client selection via submodular optimization. *ACM Transactions on Sensor Networks*, 20(2):1–32, 2024.

[29] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[30] Xinying Ji, Jie Tian, Chaoli Sun, and Meijia Zhang. Pfed-me: Personalized federated learning based on model enhancement. In *International Conference on Intelligent Computing*, pages 263–274. Springer, 2024.

[31] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and H Brendan McMahan. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*, 2020.

[32] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.

[33] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning*, pages 5132–5143. PMLR, 2020.

[34] Priyanka Mary Mammen. Federated learning: Opportunities and challenges. *arXiv preprint arXiv:2101.05428*, 2021.

[35] Betul Yurdem, Murat Kuzlu, Mehmet Kemal Gullu, Ferhat Ozgur Catak, and Maliha Tabassum. Federated learning: Overview, strategies, applications, tools and future directions. *Heliyon*, 10(19), 2024.

[36] Tatjana Legler, Vinit Hegiste, Ahmed Anwar, and Martin Ruskowski. Addressing heterogeneity in federated learning: Challenges and solutions for a shared production environment. *Procedia Computer Science*, 253:2831–2840, 2025.

[37] Muhammad Babar, Basit Qureshi, and Anis Koubaa. Investigating the impact of data heterogeneity on the performance of federated learning algorithm using medical imaging. *Plos one*, 19(5):e0302539, 2024.

[38] Jie Wen, Zhixia Zhang, Yang Lan, Zhihua Cui, Jianghui Cai, and Wensheng Zhang. A survey on federated learning: challenges and applications. *International journal of machine learning and cybernetics*, 14(2):513–535, 2023.

[39] Yichen Li, Yuying Wang, Jiahua Dong, Haozhao Wang, Yining Qi, Rui Zhang, and Ruixuan Li. Resource-constrained federated continual learning: What does matter? *arXiv preprint arXiv:2501.08737*, 2025.

[40] Pengfei Guo, Warren Richard Morningstar, Raviteja Vemulapalli, Karan Singhal, Vishal M Patel, and Philip Andrew Mansfield. Towards federated learning under resource constraints via layer-wise training and depth dropout. *arXiv preprint arXiv:2309.05213*, 2023.

[41] Tuo Zhang, Lei Gao, Chaoyang He, Mi Zhang, Bhaskar Krishnamachari, and A Salman Avestimehr. Federated learning for the internet of things: Applications, challenges, and opportunities. *IEEE Internet of Things Magazine*, 5(1):24–29, 2022.

[42] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K Leung, Christian Makaya, Ting He, and Kevin Chan. Adaptive federated learning in resource constrained edge computing systems. *IEEE journal on selected areas in communications*, 37(6):1205–1221, 2019.

[43] Daliang Li and Junpu Wang. Fedmd: Heterogenous federated learning via model distillation. *arXiv preprint arXiv:1910.03581*, 2019.

[44] Hongyan Chang, Virat Shejwalkar, Reza Shokri, and Amir Houmansadr. Cronus: Robust and heterogeneous collaborative learning with black-box knowledge transfer. *arXiv preprint arXiv:1912.11279*, 2019.

[45] Sundeep Teki. Knowledge distillation: Principles, algorithms, applications. *Neptune.ai Blog*, 2023. Last updated: September 29, 2023.

[46] Xiaoyi Pang, Jiahui Hu, Peng Sun, Ju Ren, and Zhibo Wang. When federated learning meets knowledge distillation. *IEEE Wireless Communications*, 31(5):208–214, 2024.

[47] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[48] Neelkamal Bhuyan and Sharayu Moharir. Multi-model federated learning. In *2022 14th International Conference on COMmunication Systems & NETworkS (COMSNETS)*, pages 779–783. IEEE, 2022.

[49] Enmao Diao, Jie Ding, and Vahid Tarokh. Heterofl: Computation and communication efficient federated learning for heterogeneous clients. *arXiv preprint arXiv:2010.01264*, 2020.

[50] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. *Advances in neural information processing systems*, 33:2351–2363, 2020.

[51] Chuanguang Yang, Xinqiang Yu, Zhulin An, and Yongjun Xu. Categories of response-based, feature-based, and relation-based knowledge distillation. In *Advancements in knowledge distillation: towards new horizons of intelligent systems*, pages 1–32. Springer, 2023.

[52] Chandra Thapa, Pathum Chamikara Mahawaga Arachchige, Seyit Camtepe, and Lichao Sun. Splitfed: When federated learning meets split learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 8485–8493, 2022.

[53] Zohra Dakhia and Massimo Merenda. Client selection in federated learning on resource-constrained devices: A game theory approach. *Applied Sciences*, 15(13):7556, 2025.

[54] Guanyu Ding, Zhengxiong Li, Yusen Wu, Xiaokun Yang, Mehrdad Aliasgari, and Hailu Xu. Towards an efficient client selection system for federated learning. In *International Conference on Cloud Computing*, pages 13–21. Springer, 2022.

[55] Ala Gouissem, Zina Chkirbene, and Ridha Hamila. A comprehensive survey on client selections in federated learning. *Innovation and Technological Advances for Sustainability*, pages 417–428, 2024.

[56] Yuxin Shi, Zelei Liu, Zhuan Shi, and Han Yu. Fairness-aware client selection for federated learning. In *2023 IEEE international conference on multimedia and expo (ICME)*, pages 324–329. IEEE, 2023.

[57] Yae Jee Cho, Jianyu Wang, and Gauri Joshi. Client selection in federated learning: Convergence analysis and power-of-choice selection strategies. *arXiv preprint arXiv:2010.01243*, 2020.

[58] Rana Albelaihi, Liangkun Yu, Warren D Craft, Xiang Sun, Chonggang Wang, and Robert Gazda. Green federated learning via energy-aware client selection. In *GLOBECOM 2022-2022 IEEE Global Communications Conference*, pages 13–18. IEEE, 2022.

[59] Milad Rahmati. Energy-aware federated learning for secure edge computing in 5g-enabled iot networks. *Journal of Electrical Systems and Information Technology*, 12(1):13, 2025.

[60] Yae Jee Cho, Samarth Gupta, Gauri Joshi, and Osman Yağan. Bandit-based communication-efficient client selection strategies for federated learning. *arXiv preprint arXiv:2012.08009*, 2020.

[61] Zhe Qu, Rui Duan, Lixing Chen, Jie Xu, Zhuo Lu, and Yao Liu. Context-aware online client selection for hierarchical federated learning. *IEEE Transactions on Parallel and Distributed Systems*, 33(12):4353–4367, 2022.

[62] Monalisa Panigrahi, Sourabh Bharti, and Arun Sharma. Feddcs: A distributed client selection framework for cross device federated learning. *Future Generation Computer Systems*, 144:24–36, 2023.

[63] Fernanda Famá, Charalampos Kalalas, Sandra Lagen, and Paolo Dini. Measuring data similarity for efficient federated learning: a feasibility study. *arXiv preprint arXiv:2403.07450*, 2024.

[64] Tuong Minh Nguyen, Kim Leng Poh, Shu-Ling Chong, and Jan Hau Lee. Feddss: A data-similarity approach for client selection in horizontal federated learning. *International Journal of Medical Informatics*, 192:105650, 2024.

[65] Lei Fu, Huanle Zhang, Ge Gao, Mi Zhang, and Xin Liu. Client selection in federated learning: Principles, challenges, and opportunities. *IEEE Internet of Things Journal*, 10(24):21811–21819, 2023.

[66] Carl Smestad and Jingyue 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*, pages 2–11, 2023.

[67] Yousef Alsenani, Rahul Mishra, Khaled R Ahmed, and Atta Ur Rahman. Fedsikd: Clients similarity and knowledge distillation: Addressing non-iid and constraints in federated learning. *arXiv preprint arXiv:2402.09095*, 2024.

[68] Majid Morafah and Mahdi Morafah. Clustered federated learning: A review. *Federated Learning-A Systematic Review*, 2025.

[69] Minghao Li, Dmitrii Avdiukhin, Rana Shahout, Nikita Ivkin, Vladimir Braverman, and Minlan Yu. Federated learning clients clustering with adaptation to data drifts. *arXiv preprint arXiv:2411.01580*, 2024.

[70] Christopher Briggs, Zhong Fan, and Peter Andras. Federated learning with hierarchical clustering of local updates to improve training on non-iid data. In *2020 international joint conference on neural networks (IJCNN)*, pages 1–9. IEEE, 2020.

[71] Md Sirajul Islam, Simin Javaherian, Fei Xu, Xu Yuan, Li Chen, and Nian-Feng Tzeng. Fedclust: Optimizing federated learning on non-iid data through weight-driven client clustering. In *2024 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 1184–1186. IEEE, 2024.

[72] Guixun Luo, Naiyue Chen, Jiahuan He, Bingwei Jin, Zhiyuan Zhang, and Yidong Li. Privacy-preserving clustering federated learning for non-iid data. *Future Generation Computer Systems*, 154:384–395, 2024.

[73] Yashothara Shanmugarasa, Hye-young Paik, Salil S Kanhere, and Liming Zhu. A systematic review of federated learning from clients' perspective: challenges and solutions. *Artificial Intelligence Review*, 56(Suppl 2):1773–1827, 2023.

[74] Federico Lucchetti, Jérémie Decouchant, Maria Fernandes, Lydia Y Chen, and Marcus Völp. Federated geometric monte carlo clustering to counter non-iid datasets. *arXiv preprint arXiv:2204.11017*, 2022.

[75] Daniel M Jimenez G, David Solans, Mikko Heikkila, Andrea Vitaletti, Nicolas Kourtellis, Aris Anagnostopoulos, and Ioannis Chatzigiannakis. Non-iid data in federated learning: A systematic review with taxonomy, metrics, methods, frameworks and future directions. *arXiv e-prints*, pages arXiv–2411, 2024.

[76] Ignacy Stępka, Nicholas Gisolfi, Kacper Trębacz, and Artur Dubrawski. Mitigating persistent client dropout in asynchronous decentralized federated learning. *arXiv preprint arXiv:2508.01807*, 2025.

[77] MD Nasim, Fatema Tuj Johura Soshi, Parag Biswas, AS Ferdous, Abdur Rashid, Angona Biswas, and Kishor Datta Gupta. Principles and components of federated learning architectures. *arXiv preprint arXiv:2502.05273*, 2025.

[78] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[79] Jay Nandy, Wynne Hsu, and Mong Li Lee. Towards maximizing the representation gap between in-domain & out-of-distribution examples. *Advances in neural information processing systems*, 33:9239–9250, 2020.

[80] Aviral Kumar Goyal, Manish Pandey, Dhirendra Pratap Singh, Jaytrilok Choudhary, and Rahul Haripriya. Fedcontrast: A contrastive learning framework to mitigate client drift under statistical heterogeneity in federated multi-class soil classification. 2025.

[81] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. *Advances in neural information processing systems*, 33:19586–19597, 2020.

[82] AI-Benchmark. Ai-benchmark ranking. https://ai-benchmark.com/ranking.html, 2025. Accessed: September 24, 2025.

[83] Goran Kalic, Iva Bojic, and Mario Kusek. Energy consumption in android phones when using wireless communication technologies. In *2012 Proceedings of the 35th International Convention MIPRO*, pages 754–759. IEEE, 2012.

[84] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study. In *2022 IEEE 38th international conference on data engineering (ICDE)*, pages 965–978. IEEE, 2022.

Appendix A

Data Heterogeneity Simulation and Client grouping

A.1 Dirichlet non-IID data partitioning

For label, features and quality skew simulation of multiple clients (federated datasets), various methods were used for creating synthetic partitions of a centralized dataset.

Label distribution skew is when the label distributions vary across clients. Example, some hospitals are more specialized in several specific kinds of diseases and have more records on them. To simulate label distribution skew, there are different label imbalance settings quality-based label imbalance, distribution-based label imbalance (here we are focusing on this).

Distribution based label imbalance: each party is allocated a proportion of the sample of each label according to Dirichlet distribution, that is it uses Dirichlet distribution to partition the data. The Dirichlet distribution models the probabilities assigned to each label. An advantage of this approach is that we can flexibly change the imbalance level by varying the concentration parameter Beta. For instance, if Beta is set to smaller value, then the partition is more unbalanced and more distinct distributions across clients, while higher value results in more similar distributions. On below Figure it is shown that how the data partition using Dirichlet distribution for MNIST dataset is performed of the given Party ID (client devices). here on the below Figure A.1 it is illustrated how data is partitioned using Dirichlet distribution [84].

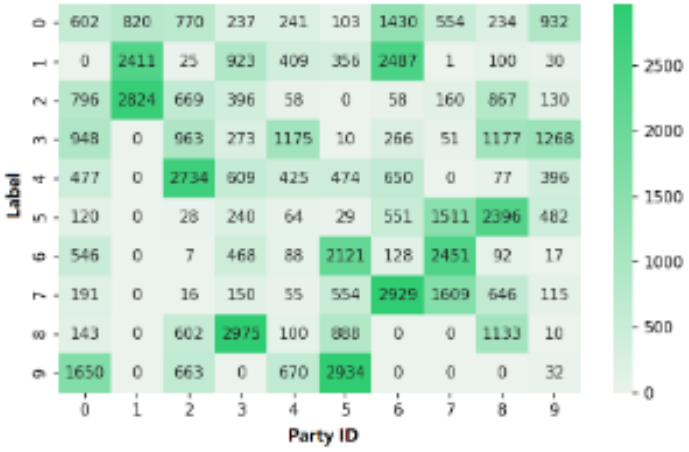


Figure A.1: Here it is shown distribution-based label imbalance on MNIST dataset with Beta = 0.5, each rectangle value shows the number of data samples of each class (0-9) belonging to specific partition.

A.2 Client data similarity calculation

We have used the mean and standard deviation of a client data statistics distribution for client similarity calculation, and they are key statistical measures that describe the characteristics of data distribution.

The mean for a client means the mean of its local data distribution such as, on the MNIST image dataset it is the average pixel intensity. So, clustering by mean groups clients whose local data is centered around a similar point, ensures that clients with similar data content are grouped together. On the other hand, the standard deviation for a client indicates how diverse or uniform its local dataset. Clustering based on standard deviation groups clients' that have similar levels of diversity in their data.

A.3 Client clustering using silhouette coefficient and Davis Bouldin

The optimal number of clusters K in the K-means clustering, is typically found by using metrics namely silhouette coefficient and Davis Bouldin index. The sever employ those metrics to evaluate the quality of clustering and determining the appropriate K. for having strong approach, we have used the two metrics in conjunction to one another.

silhouette coefficient is internal validation metric that measure two key aspects of clustering. Cohesion, which quantifies how close data points with in a cluster are to each other and separation, assesses how distinct clusters are from one another. It serves as an internal basis to assess clustering quality by measuring clustering cohesion and separation, which is applicable to many clustering algorithms like K-means clustering here as our scenario. silhouette coefficient is computed as the difference between the average distance to points in the nearest cluster and the average distance to points in the same cluster, divided by maximum of the two values. The score ranges from -1 to +1 where, values close to +1 indicate well matched samples with their clusters, values near 0 suggest samples on cluster boundaries and negative values imply possible misclassifications.

The Davis Bouldin index, measures average similarity ratio between each cluster and its most similar neighbor. Similarity here means the ratio of with in cluster scatter to between cluster distance. Lower value is better, indicating that clusters are more spread out from each other and are more compact internally. This complementary metric to the above can provide additional perspectives on clustering performance.