Asymmetric Clustering in Federated Continual Learning

by

Zehao Zhang

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2024

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Asymmetric clustering represents a critical yet under-explored challenge in Clustered Federated Learning (CFL). Existing methods often compromise data utilization or model accuracy by either separating devices with different data quality into distinct clusters or merging all devices into a single cluster. The need for asymmetric clustering arises in practical scenarios where not all devices contribute equally due to varying data quality or quantity. For example, in healthcare, devices at a research hospital might generate high-quality medical imaging data compared to a small clinic. Asymmetric clustering allows high-quality data sources to enhance the learning of models on devices with lower-quality data without the need for reciprocity, which is crucial in such imbalanced environments. We introduce a novel federated learning technique that enables selective contributions from some devices to others' model training without requiring equal give-and-take. Crucially, our approach excels in the Federated Continual Learning (FCL) setting by addressing temporal heterogeneity and concept drift through its ensemble features. Through detailed empirical evaluations, we validate that our approach not only efficiently generates high-quality asymmetric clustering but also significantly enhances performance in continual learning settings. This adaptability makes it highly suitable for real-world applications where data distributions are not static but evolve over time.

Acknowledgments

I would like to thank my supervisor Pascal Poupart for his constant support, insightful advice, and patient guidance throughout my entire Masters degree. His expertise and mentorship have been pivotal to my academic and personal growth.

I would also like to thank my colleagues at Huawei Noah's Ark Lab: Guojun Zhang, Xi Chen, Kaiyang Guo, Mohsin Hasan, Ahmad Rashid and Haolin Yu. Their innovative ideas and valuable feedback have significantly enriched my research experience and contributed to the development of this work.

I would like to thank my committee members Hong Zhang, Olga Veksler for expert advice and constructive feedback regarding my work.

I would like to thank my family – my parents, siblings, and the great-aunt and her family – for their unconditional love and support during my studies abroad.

Lastly, I would like to thank my girlfriend for her companionship and encouragement during this challenging yet rewarding journey.

Dedication

This is dedicated to my beloved mom.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Federated learning [12] is a machine learning technique designed to train algorithms across decentralized devices while keeping data localized, thus addressing privacy, security, and data access challenges. Unlike traditional centralized machine learning methods where all data is uploaded to one server, federated learning allows for the model to be brought to the data source where training occurs. This approach is particularly valuable in scenarios where data privacy is important, such as in healthcare, finance, and mobile computing. For example, smartphones that utilize predictive text input features can improve their models using federated learning by learning from user interactions without ever needing to upload individual typing data to a central server. However, federated learning introduces complexities such as handling non-IID (independently and identically distributed) data across various devices, dealing with devices that have varying computational and storage capacities, and managing communication costs and efficiencies.

In practice, it is common for devices to encounter data from diverse distributions. Since heterogeneous data may induce different optimal predictors at different devices, this has led to the development of personalized federated learning techniques [6]. A popular approach consists of training a global predictor that is adapted or fine-tuned for each device. However, this assumes that the optimal predictors at each device are similar enough that fine-tuning / adapting a global predictor will be sufficient. In cases where some optimal predictors are very different and fine-tuning is insufficient, then clustered federated learning [15] becomes attractive. For instance, in mobile keyboard prediction, where users from different regions have distinct linguistic preferences and slang, fine-tuning a single global model may not be effective; clustered federated learning, on the other hand, allows for creating separate models for different linguistic groups to ensure that predictions remain relevant and accurate. In clustered FL, devices are partitioned in clusters such that devices share models only with

the other devices in their cluster. When clusters combine devices with similar data while making sure that devices with very different data are in different clusters, then learning will be more effective. Existing techniques for clustered FL [15] can learn clusters dynamically. However, most existing techniques assume a fixed number of clusters that is known a priori and all existing techniques assume that each device contributes to a single cluster. We describe a technique that relaxes those two assumptions.

In traditional clustering, determining the correct number of clusters beforehand can be challenging. This is particularly true in federated learning environments where data is distributed across numerous devices with potentially diverse data distributions. Static clustering methods that assume a fixed number of clusters can lead to inefficiencies and inaccuracies, as they might not accommodate the dynamic nature of real-world data, which can vary in terms of volume, variety across different devices.

Adaptive clustering addresses this limitation by employing algorithms that dynamically adjust the number of clusters based on the evolving characteristics of the data. Instead of pre-defining a cluster count, adaptive clustering methods continuously analyze the incoming data and modify the cluster count in real-time. This flexibility allows the learning process to maintain high levels of efficiency and adaptability.

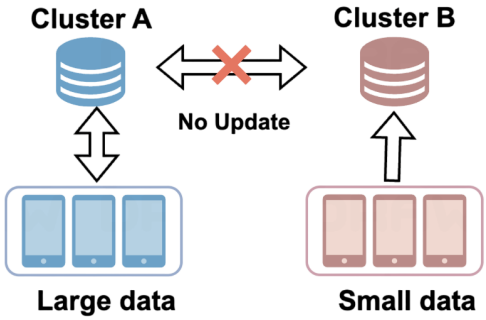

Figure 1.1: Symmetric Clustering

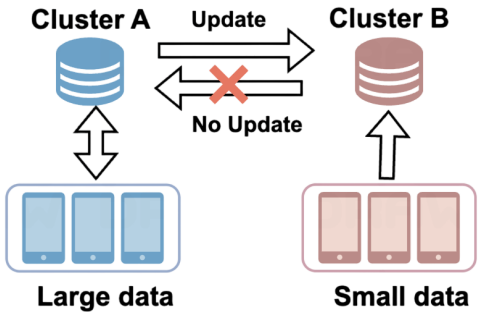

Figure 1.2: Asymmetric Clustering

In federated learning, asymmetric scenarios often arise where the benefits of model sharing are not reciprocal between devices. For instance, consider a situation involving two devices, device A and device B. Device A has a large dataset characterized by the underlying conditional distribution $p_A(y \mid x)$, whereas Device B has a smaller dataset with a similar conditional distribution $p_B(y \mid x)$ that matches $p_A(y \mid x)$ for 90% of the input x.

For device A, incorporating device B's data could potentially introduce a bias that

might degrade the accuracy of its own model because of the 10% divergence in their data distributions. Thus device A would not wish to train on data from device B. On the other hand, for device B, clustering with device A could significantly reduce variance owing to the greater volume of data it would have access to, thereby enhancing its overall performance. Despite the risk of introducing some bias, device B's primary concern is to mitigate variance due to its limited data.

In addition to addressing static clustering challenges, our clustering approach can also handle the continual learning setting. Continual learning in the context of federated learning involves updating the model as new data arrives from various devices, which may exhibit changing data characteristics due to evolving user behaviors. Our approach effectively manages this by dynamically adjusting the clusters based on the incoming data stream, ensuring that the model remains relevant and accurate over time. This capability is crucial in real-world applications where data evolves due to changes in user behavior, seasonal effects, or emerging trends. The ability to handle both spatial and temporal heterogeneity makes our approach particularly effective in environments where data distributions are not only uneven across devices but also change over time.

1.1 Contributions

The main contributions of this work are outlined as follows:

- **Introduction of Asymmetric Clustering**: We introduce a novel concept called asymmetric clustering, which allows for more flexible and dynamic cluster formations. This approach better accommodates the varying quality and distribution of data among devices, enhancing the applicability and efficacy of federated learning models.
- **Development of Dynamic Clustering**: Our research advances the implementation of adaptive clustering within the existing frameworks of clustered federated learning. Our method allows clusters to be initially formed based on real-time data observations and dynamically adjusted as data characteristics evolve.
- **Handling Spatial and Temporal Data Heterogeneity**: Our method, termed C-FAACL (Clustered Federated Asymmetric Adaptive Clustering Learning), adeptly manages both spatial and temporal heterogeneity in federated environments. This capability is crucial in scenarios where data distributions vary not only across devices but also over time—due to factors like seasonal changes or shifts in user behavior—making C-FAACL well-suited for real-world applications.

- **Empirical Validation**: We conduct extensive experiments to validate the performance of both FAACL and C-FAACL, demonstrating their competitive advantage over traditional federated learning baselines. Our results confirm the effectiveness and scalability of our methods in a variety of federated settings, establishing their practical viability.

1.2 Thesis Outline

This thesis is organized into five chapters, below is a summary of each chapter:

- **Chapter 2 Background**: This chapter provides a foundational overview of the terms and notations that will be used throughout the thesis. It also reviews existing literature on clustered federated learning and federated continual learning, establishing the context for the subsequent chapters and highlighting the gaps that this research aims to fill.
- **Chapter 3 The FAACL Method**: The Federated Asymmetric Adaptive Clustering Learning (FAACL) method is introduced and explored in depth in this chapter. It details the algorithmic framework and discusses various experiments conducted to assess the method's efficacy and robustness. This chapter serves as a comprehensive guide to understanding how FAACL operates within standard federated learning settings.
- **Chapter 4 The C-FAACL Method**: Chapter 4 expands upon the concepts introduced in Chapter 3 by adapting the FAACL method to a continual learning context, referred to as C-FAACL. This adaptation addresses the challenges of applying federated learning over continuous data streams. The chapter details the algorithm modifications necessary for this adaptation and presents experimental results that validate the effectiveness of C-FAACL in dynamic environments.
- **Chapter 5 Conclusion and Future Work**: The final chapter concludes the thesis by summarizing the key findings and contributions of the research. It discusses the limitations of the current methods and proposes potential direction for future research.

Chapter 2

Background

This chapter establishes the foundational framework essential for understanding the later chapters of the thesis. In the Related Work section, we provide a brief overview of the existing research in Clustered Federated Learning (CFL) and Federated Continual Learning (FCL). We discuss significant contributions, methodologies, and the notable limitations of current approaches. The Notation section introduces and clarifies the notation that will be consistently used throughout the thesis.

2.1 Related Work

2.1.1 Clustered Federated Learning

Clustered Federated Learning (CFL) represents a significant advancement in managing distributed data across various devices. This subsection reviews key methodologies and their respective contributions to the field.

- **Iterative Federated Clustering Algorithm (IFCA)** [7] starts with a predefined number of cluster models at the server. Devices determine their cluster identity based on which models minimize their local loss.
- **Federated Stochastic Expectation Maximization (FeSEM)** [19] begins with fixed number of clusters and iteratively assigns devices to the nearest cluster based on the $L2$ distance of the model parameters. Each cluster updates its model by averaging the models of the assigned devices.

- **FedGroup** [5] clusters devices according to the cosine similarity of their gradients and facilitates both inter-cluster and intra-cluster training alongside device migration.
- **FedSoft** [14] operates similarly to IFCA but introduces flexibility by allowing devices to belong to multiple clusters. Each cluster's importance is determined based on the local loss for each data point.
- **FedDrift** [8] is designed for continual learning settings. It starts with an assumption of homogeneous data distribution but can adapt to changes by initiating new clusters when significant shifts in data distribution are detected through loss comparison.

Despite these advancements, two primary limitations persist in the current CFL:

- **Fixed Number of Clusters**: Except for FedDrift, most of the previous approaches use a fixed number of clusters for device grouping. This fixed cluster count presents a challenge, as it requires certain prior knowledge and needs to be accurate. If the initial guess for the number of clusters is too low, devices with varying data distributions may be incorrectly grouped together, leading to suboptimal predictors for those devices. Conversely, an excessive number of clusters can scatter devices with similar data distributions across different clusters, resulting in suboptimal predictors due to a reduced amount of data for cluster model training.
- **Symmetric Clustering Limitations**: There is no natural extension from symmetric clustering to asymmetric clustering. Devices either support each other or they do not. Traditional CFL approaches typically restrict each device to a single cluster (e.g., IFCA, FeSEM), or, as seen in soft clustering methods like FedSoft, allow devices to influence multiple clusters without adequately considering the overall impact on cluster integrity. In environments where data quality varies considerably, such strategies may compromise the robustness of clusters initially dominated by high-quality data, thus failing to balance individual benefits with collective goals effectively.

2.1.2 Federated Continual Learning

Federated Continual Learning (FCL) [20] tackles the challenges of learning continuously over distributed and heterogeneous datasets while preserving the privacy and autonomy of each participating device. Various strategies have been explored, categorized into regularization methods, parameter isolation, and clustering methods.

- **Regularization Methods**: Regularization approaches such as Elastic Weight Consolidation (EWC) [9] are designed to mitigate catastrophic forgetting by penalizing significant changes to parameters critical for previously learned tasks. Although effective in environments with gradual task transitions, these methods often face challenges in federated settings where device data distributions are markedly diverse.

- **Parameter Isolation**: Parameter isolation techniques, such as Adaptive Parameter Decomposition (APD) [22] manage task-specific model parameters to prevent the overwriting of previously acquired knowledge by new tasks. Integrating attention mechanisms with parameter isolation strategies can dynamically adjust the influence of these isolated parameters, enhancing model personalization across devices. However, deriving attention scores from model parameters rather than direct data features—often a necessity due to privacy concerns—can complicate the accurate representation of each device's unique data distribution.

- **Clustering Methods**: Clustering approaches like FedDrift [8] dynamically group similar devices into clusters and reassess device assignments as new data emerges. However, these methods typically restrict devices to contribute to only a single cluster model, limiting their effectiveness in scenarios where device data profiles are asymmetric or evolve uniquely, thereby necessitating more flexible or multiple cluster memberships to accurately reflect changes in device data distributions.

2.2 Notation

In this section, we establish the notation that will be consistently used throughout this thesis.

Device and Data: Consider a set of n devices denoted as $\mathcal{D} = \{d_1, ..., d_n\}$. For each device d in this set, its associated dataset is denoted by Z_d. Each data point within this dataset, represented as $z = (x, y)$, is assumed to be sampled from an underlying distribution, which we denote as $P_d(z)$, where $z \in Z_d$. Additionally, we partition Z_d into three subsets: the training set Z_d^{train}, the validation set Z_d^{val}, and the test set Z_d^{test}.

Cluster Structure and Clustering: We define C_j as the j^{th} cluster, which consists of three primary components:

- **Device Set $\mathbf{C_j}$.D**: This set includes the devices that directly contribute to the cluster.

- **Supportive Clusters $\mathbf{C_j}$.sup**: These are clusters that establish inter-cluster relationships. For instance, if cluster C_A requires support from another cluster, C_B, C_B is considered a supportive cluster to C_A, indicated by $C_B \in C_A.sup$.
- **Cluster Model Parameters $\mathbf{C_j}.\theta$**: The model parameters associated with each cluster.

While both $C_j.D$ and supportive clusters in $C_j.sup$ contribute to training the model $C_j.\theta$, only the devices in $C_j.D$ utilize model $C_j.\theta$ for prediction. This differentiation allows for enhanced data utilization and model accuracy through collaborative but non-intrusive inter-cluster support.

A clustering is defined as a collection of clusters, denoted by $\mathcal{C}$, where a potential clustering of size k for a set of devices $\mathcal{D}$ might be represented as $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$. Each individual device set $C_i.D$ collectively covers $\mathcal{D}$ without overlap.

Continual Learning and Memory structure: In the continual learning setting, we define T as the total number of time steps. At each time step t, every device d_i receives a new dataset denoted by $Z_d^{(t)}$ (also divided into $Z_d^{train_{(t)}}, Z_d^{test_{(t)}}, Z_d^{val_{(t)}}$). The clustering outcome at each time step t is represented by $\mathcal{C}^{(t)}$, where $C_j^{(t)}$ refers to the j^{th} cluster at that specific time.

To effectively manage temporal data, we introduce a memory structure, denoted as $\mathcal{M}$. Unlike a traditional replay buffer which is used to sample data for model updates, $\mathcal{M}$ serves as a passive repository. It stores historical data crucial for maintaining data diversity and providing a historical context, essential for detecting and handling concept drifts. For each device d_i at time t, the memory $d_i.\mathcal{M}^{(t)}$ stores a tuple $\{S, C\}$, where S is a sample of local data, and C is the device's cluster assignment at that time step. Similarly, for each cluster C_j at time t, the memory $C_j.\mathcal{M}^{(t)}$ contains records of the set of devices $\mathcal{D}_j$ assigned to the cluster C_j. This structured approach to memory utilization ensures that each device and cluster has access to its relevant historical data, supporting more informed and context-aware decisions in the clustering and learning processes.

Chapter 3

Federated Adaptive Asymmetric Clustered Learning

This chapter introduces the Federated Asymmetric Adaptive Learning (FAACL) method. We begin with a problem statement in Section 3.1, a detailed algorithm in Section 3.2, and the experiments in Section 3.3.

3.1 Problem Statement

Consider the loss function $\ell(\theta, z)$ of the model parameterized by θ on data point $z = (x, y)$. Our primary objective is to construct a clustering $\mathcal{C}$ that minimizes the population loss $L(\mathcal{C})$ represented as follows.

$$L(\mathcal{C}) = \sum_{C \in \mathcal{C}} \sum_{d \in C.D} E_{z \sim P_d(z)}[\ell(C.\theta, z)] \tag{3.1}$$

This equation encapsulates the total loss across all clusters within the clustering $\mathcal{C}$, where each cluster's contribution to the loss is determined by its assigned model parameters and the data from devices within that cluster.

Unlike previous works, our approach does not rely on prior knowledge about the number of clusters. Consequently, we cannot start with a predetermined number of clusters and simply minimize their losses to obtain optimal parameters $\mathcal{C}^* = \text{argmin}_{\mathcal{C}}\ L(\mathcal{C})$. To address this challenge, we have developed a comprehensive methodology containing several key components:

1. **Cluster Initialization:** We describe a technique for the initial formation of clusters. The details of this process will be elaborated in Section 3.2.1.

2. **Cluster Merge:** We introduce criteria for the aggregation of clusters, enhancing the adaptability and efficiency of our model. This will be explained in Section 3.2.2.

3. **Cluster Training:** This aspect involves strategies for effectively training within each cluster, which will be discussed in Section 3.2.3.

4. **Proposed Federated Clustering Framework:** Our research culminates in a federated clustering framework designed to facilitate adaptive and asymmetric clustering. This comprehensive framework will be detailed in Section 3.2.4.

5. **Complexity Analysis:** An in-depth analysis of the complexity of our proposed framework is provided, along with the necessary proofs in Section A.

3.2 Algorithm Details

3.2.1 Clustering Initialization

The initialization phase of our FAACL algorithm sets up an initial clustering for the cold-start system. The clustering initialization begins with the server distributing a set of random parameters, θ, to all devices in the device set $\mathcal{D}$. Each device d_i forms a singleton cluster that includes only itself, i.e., $C_i.D = \{d_i\}$. Each device trains the model with initial parameters θ locally with its training data, aiming to minimize the loss function $l(\theta, z)$ over its data points. The constructed clusters are sent back to the server. The server collects all individual clusters to form the initial clustering $\mathcal{C}$, which consists of n singleton clusters. The process is systematically described in the Algorithm 1 and is illustrated in Figure 3.1.

Algorithm 1 Clustering Initialization

Input: Device set of size n, $\mathcal{D} = \{d_1, \ldots, d_n\}$
Output: Initial clustering $\mathcal{C}$

Server: Initialize $\mathcal{C} = \{\}$ and random parameters θ
Server: Distribute θ to all devices in $\mathcal{D}$
for each device $d_i \in \mathcal{D}$ **do**
  Device: Form a new cluster C_i with $C_i.D = \{d_i\}$, $C_i.sup = \{\}$
  Device: Initialize parameters $C_i.\theta = \theta$ and $C_i.\theta = argmin_\theta \sum_{z \in Z_{d_i}^{train}} \ell(\theta, z)$
  Device: Send the cluster C_i to server
Server: $\mathcal{C} \leftarrow \{C_i\}_{i=1}^n$
Return Initial clustering $\mathcal{C}$

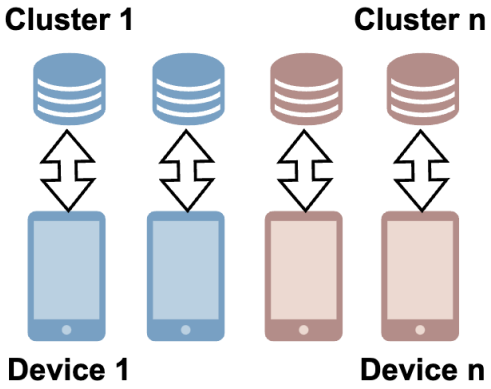

Figure 3.1: Clustering Initialization

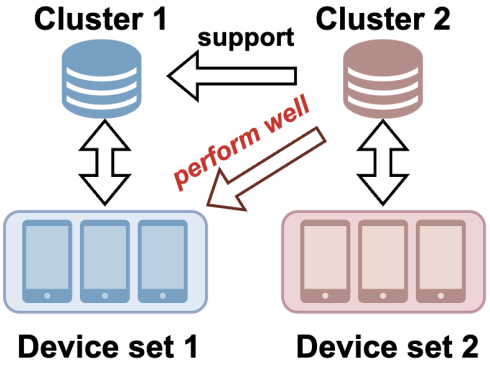

Figure 3.2: Cluster Support

3.2.2 Cluster Merge

Following the initialization phase that results in n singleton clusters, the next stage of our methodology involves an iterative process to merge similar clusters. This is achieved by first updating each cluster's supportive connections by exchanging cluster models among devices from different clusters as shown in Algorithm 2. Then we aggregate pairs of clusters that are supportive clusters of each other, as described in Algorithm 3.

Model Comparison and Support Determination: To decide whether cluster C_2 is a supportive cluster for cluster C_1 depends on the performance of their respective model

parameters $C_2.\theta$ and $C_1.\theta$ across all devices in $C_1.D$ (illustrated in Figure 3.2). If the model parameters of C_2 are found to be no worse than those of C_1 for all devices in C_1, C_2 is added to the supportive clusters of C_1.

To compare the performances of $C_1.\theta$ and $C_2.\theta$ for a given device d, losses are computed for each data point in the validation set Z_d^{val}. This allows for a direct comparison of the two models on the same dataset. To objectively evaluate their performance, statistical tests, such as the Wilcoxon signed-rank test [13], are applied. The Wilcoxon signed-rank test is a non-parametric method used to compare paired-samples and assess whether their distributions differ. It serves as an alternative to the paired Student's t-test [23] and does not require the assumption that the distribution is normally distributed. Specifically, let $\ell(\theta_{C_1}, z_d^{val})$ and $\ell(\theta_{C_2}, z_d^{val})$ denote the losses of θ_{C_1} and θ_{C_2} respectively on a data point $z \in Z_d^{val}$. The $Wilcoxon(L, \epsilon)$ test with the null hypothesis formulated as $H_0 : \ell(C_1.\theta, z) + \epsilon < \ell(C_2.\theta, z)$, with ϵ representing the threshold, L denoting the set of losses by θ_{C_1} and θ_{C_2} on a dataset and α denoting the preset significance level.

If the statistical test yields a p-value smaller than α, it indicates strong evidence against the null hypothesis, suggesting that the model parameters $C_1.\theta$ are not more effective than $C_2.\theta$. Consequently, it can be concluded that device d might benefit from the model of cluster C_2, thus the $support\_flag$ remains True until some p-value exceeds α. Following the assessment by other devices in C_1, if all devices agree that they can benefit from the model of cluster C_2, C_2 is then added to the supportive clusters of C_1.

Merging Clusters: Upon establishing that clusters are mutually supportive, they are merged into a new cluster as part of the process detailed in Algorithm 3. In this newly formed cluster, the device sets are combined, creating a union of the devices from the previous clusters. Importantly, rather than averaging the model parameters, the model for the new cluster is selected randomly from one of the pre-existing models. This decision is a deliberate deviation from approaches like the Federated Averaging (FedAvg) algorithm, where model parameters are frequently averaged to maintain synchronization across all participating clients. In FedAvg, the global model is typically broadcasted to clients at each communication round, which may occur after only one or a few gradient updates, ensuring consistent model performance across devices. In contrast, our clusters may have been trained independently for many iterations before the decision to merge is made. This prolonged independent training can lead to significant divergences in the model parameters, even when models are trained on data from similar distributions. Such divergences can arise due to inherent symmetries in the training process, where different but equivalent parameter sets are found as solutions by different clusters. Simply averaging these parameters upon merging could potentially dilute the distinct advantages each model has developed, leading to a loss in performance.

The detailed steps of this merging procedure are outlined in the pseudo-code provided in Algorithm 3 (see Figure 3.3).

Algorithm 2 Cluster Support

Input: Clustering $\mathcal{C}$, significance level α, threshold ϵ
Output: Updated Clustering $\mathcal{C}$

---

for each cluster $C_1 \in \mathcal{C}$ **do**
  for each distinct cluster $C_2 \in \mathcal{C}$ **do**
    Server: Initialize $support\_flag \leftarrow True$
    for each device $d \in C_1.D$ **do**
      Server: Send parameters $C_1.\theta, C_2.\theta$ to device d
      Device: $L \leftarrow \{(\ell(C_1.\theta, z), \ell(C_2.\theta, z)) | z \in Z_d^{val}\}$
      Device: Computes $p \leftarrow Wilcoxon(L, \epsilon)$
      if $p > \alpha$ **then**
        Server: $support\_flag \leftarrow False$ and **break**
    if $support\_flag$ is True **then**
      Server: $C_1.sup.add(C_2)$
Return Updated clustering $\mathcal{C}$

Algorithm 3 Cluster Merge

Input: Clustering $\mathcal{C}$
Output: Refined Clustering $\mathcal{C}$

Server: Let $\mathcal{C}_{new}$ and $\mathcal{C}_{old}$ be empty
for each cluster $C_1 \in \mathcal{C}$ not in $\mathcal{C}_{old}$ **do**
  for each distinct cluster $C_2 \in \mathcal{C}$ not in $\mathcal{C}_{old}$ **do**
    if $C_1 \in C_2.sup$ and $C_2 \in C_1.sup$ **then**
      Server: Form new cluster C'
      $C'.D = C_1.D \cup C_2.D$, $C'.sup = C_1.sup \cap C_2.sup$, $C'.\theta = random(C_1.\theta, C_2.\theta)$
      Server: $\mathcal{C}_{new}.add(C'), \mathcal{C}_{old}.add(C_1, C_2)$
Server: $\mathcal{C} \leftarrow \mathcal{C} - \mathcal{C}_{old} + \mathcal{C}_{new}$
Return Refined clustering $\mathcal{C}$

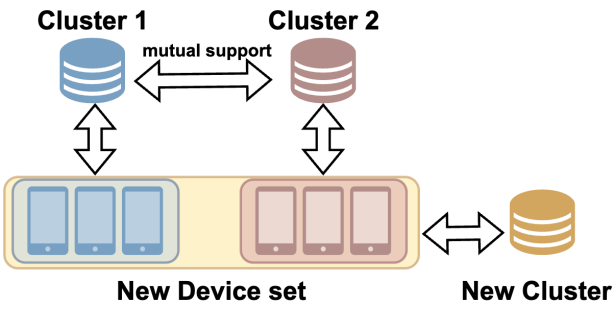

Figure 3.3: Cluster Merge

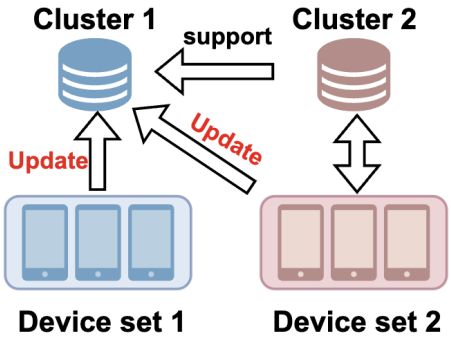

Figure 3.4: Cluster Training

One limitation of the cluster merging process is its sensitivity to the order of merging. For example, if clusters from differing distributions are merged accidentally before they are merged with the clusters from the same distribution, it could lead to degraded performance in the resulting cluster. However, this risk can be effectively managed by fine-tuning the significance level (α) of the statistical tests. A large α tends to be more permissive, potentially merging clusters that are not sufficiently similar, and a smaller α imposes a stricter criterion, potentially resulting in an excessive number of clusters. By fine-tuning α, we can minimize the likelihood of inappropriate merges.

3.2.3 Cluster Training

During the training phase, each cluster C_j engages in a series of training iterations to refine and optimize its model parameters. This optimization process is aimed at minimizing the collective loss calculated from all the data available from devices that are part of the cluster $C_j.D$ as well as data from devices belonging to supportive clusters $C_j.sup$. The steps are illustrated in Figure 3.4 and details are provided in Algorithm 4.

Algorithm 4 Cluster Train

Input: Clustering $\mathcal{C}$
Output: Updated Clustering $\mathcal{C}$

---

for each cluster $C \in \mathcal{C}$ **do**
  for each device $d \in C.D$ **do**
    Server: Send cluster model parameters $C.\theta$ to d
    Device: $C_d.\theta \leftarrow C.\theta - \lambda\nabla(\sum_{z \in Z_d^{train}} loss(C.\theta, z))$ (gradient descent)
    Device: Send $C_d.\theta$ and $|Z_d^{train}|$ to server
  for each supportive cluster $C' \in C.sup$ **do**
    for each device $d' \in C'.D$ **do**
      Server: Send cluster model parameters $C.\theta$ to d'
      Device: $C_{d'}.\theta \leftarrow C.\theta - \lambda\nabla(\sum_{z \in Z_{d'}^{train}} loss(C.\theta, z))$ (gradient descent)
      Device: Send $C_{d'}.\theta$ and $|Z_{d'}^{train}|$ to server
  Server: $C.\theta \leftarrow \frac{\sum_{d \in C.D}(|Z_d^{train}| \times C_d.\theta) + \sum_{C' \in C.sup}\sum_{d' \in C'.D}(|Z_{d'}^{train}| \times C_{d'}.\theta)}{\sum_d |Z_d^{train}| + \sum_{C' \in C.sup}\sum_{d' \in C'.D}|Z_{d'}^{train}|}$ (weighted average)
Return Updated clustering $\mathcal{C}$

3.2.4 Proposed Federated Clustering Method

In this section, we introduce two advanced approaches for federated clustering under the FAACL framework: Flat FAACL and Hierarchical FAACL. These strategies are designed to handle the clustering of devices effectively while balancing computational efficiency and clustering performance. The runtime complexity is analyzed in Appendix A.

Flat FAACL Method: Flat FAACL integrates the previous algorithms into a cohesive approach. The term "flat" in this context indicates that the clustering approach treats all devices on the same level. This flat clustering processes all devices simultaneously, directly compares and merges clusters. It starts with cluster initialization, and repeatedly train each cluster, identify support relationship and attempts to merge mutual supportive clusters. When the clustering stabilizes, further training continues with the fixed cluster allocation, shown in Algorithm 5. This approach involves extensive pairwise interactions between clusters, leading to a computational complexity of $O(n^2)$ per iteration, where n denotes the number of devices.

Hierarchical FAACL Method: To optimize the computational demands posed by Flat FAACL, we propose the Hierarchical FAACL method. This approach introduces a

tiered clustering strategy, where devices are initially grouped into smaller clusters that are progressively merged to form larger clusters. This hierarchical structure significantly reduces the computational overhead by limiting the number of direct comparisons and merges required at each stage of the process. Each level of the hierarchy forms an intermediate clustering that refines the grouping of devices, enhancing the efficiency and potentially improving the adaptability of the model to changes in device data distributions.

Algorithm 5 Flat FAACL

Input: Device set $\mathcal{D}$, significance level α, threshold ϵ, initial clustering $\mathcal{C}_{init}$, number of epochs *epochs*

Output: Optimized Clustering $\mathcal{C}$

Initialize $\mathcal{C}$={}
if $\mathcal{C}_{init}$ are given **then**
  $\mathcal{C}^*$=$\mathcal{C}_{init}$
else
  $\mathcal{C}^*$=Cluster Initialization($\mathcal{D}$)
while $\mathcal{C} \neq \mathcal{C}^*$ **do**
  Update $\mathcal{C}$←$\mathcal{C}^*$
  Train clusters $\mathcal{C}^*$←Cluster Train($\mathcal{C}^*$)
  Update cluster support $\mathcal{C}^*$←Cluster Support($\mathcal{C}^*, \alpha, \epsilon$)
  Merge clusters $\mathcal{C}^*$←Cluster Merge($\mathcal{C}^*$)
while *epochs* > 0 **do**
  Train clusters $\mathcal{C}$←Cluster Train($\mathcal{C}$)
  epochs←*epochs* $- 1$
Return Final clustering $\mathcal{C}$

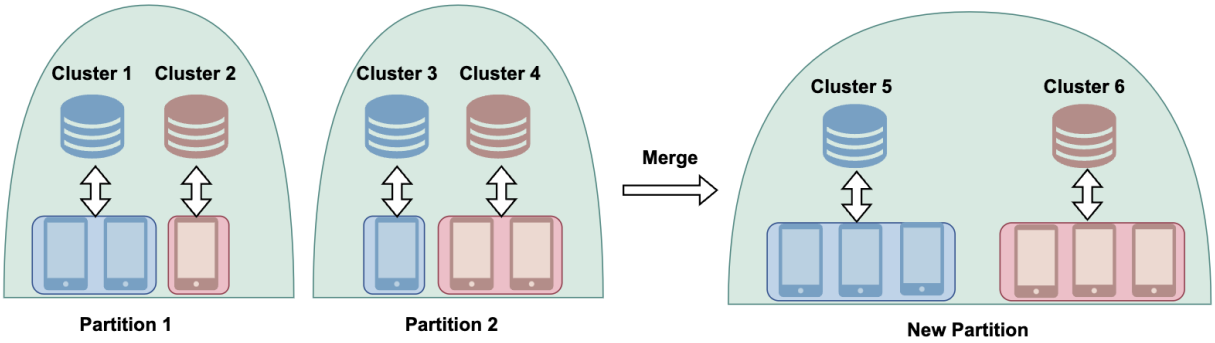

Figure 3.5: H-FAACL: Partition Merge

Partitioned Strategy for Enhanced Efficiency: To further enhance the efficiency of clustering process, we employ a strategic partitioning approach, where each partition P consists of a subset of devices and their associated clusterings.

- **Set of devices** ($\mathbf{P}.\mathbf{D}$): This subset may include devices like $\{d_1, d_2, d_3\}$, indicating the devices included in the partition.
- **Clustering formed by its device set** ($\mathbf{P}.\mathcal{C}$): Each partition also has its own clustering, such as $\{C_1, C_2\}$, where $C_1.D = \{d_1\}$ and $C_2.D = \{d_2, d_3\}$.

The partition merging process is systematically detailed in Algorithm 6, starting with n initial partitions, each containing a single device. During each iteration, two partitions, P_i and P_j are selected and merged to form a new partition P'. This new partition combines the device sets and clustering from P_i and P_j. The combined clustering $P'.\mathcal{C}$ is then used as initial parameters $\mathcal{C}_{init}$ for the subsequent application of Flat FAACL. This iterative merging continues until only one comprehensive partition remains, effectively simplifying the clustering process while aiming to retain the efficacy of the ultimate clustering outcome. The process of merging partitions and forming new clusters can be visualized in Figure 3.5.

Algorithm 6 Hierarchical FAACL

Input: Device set $\mathcal{D}$, significance level α, threshold ϵ, number of epochs *epochs*
Output: Clustering $\mathcal{C}$

Initialize partition set $\mathcal{P}$={}
for each device $d_i \in \mathcal{D}$ **do**
  Initialize partition P with $P.D$=$\{d_i\}$ and $P.\mathcal{C}$=None
  $\mathcal{P}.add(P)$
while $|\mathcal{P}| > 1$ **do**
  for each partition $P \in \mathcal{P}$ **do**
    $P.\mathcal{C}$←Flat FAACL$(P.D, \alpha, \epsilon, P.\mathcal{C}, 0)$
  Initialize a new set of partitions $\mathcal{P}^*$={}
  for P_i, P_j sampled non-repeatedly from $\mathcal{P}$ **do**
    Create new partition P' with $P'.D$=$P_i.D \cup P_j.D$ and $P'.\mathcal{C}$=$P_1.\mathcal{C} \cup P_2.\mathcal{C}$
    $\mathcal{P}^*.add(P')$
  $\mathcal{P}$←$\mathcal{P}^*$
Let P be the only remaining partition in $\mathcal{P}$, $\mathcal{C}$←$P.\mathcal{C}$
while $epochs > 0$ **do**
  Train clusters $\mathcal{C}$←Cluster Train$(\mathcal{C})$
  $epochs$←$epochs - 1$
Return Final clustering $\mathcal{C}$

3.3 Experiments

This section presents the empirical results obtained from testing our proposed FAACL methods under two different experimental scenarios—symmetric and asymmetric. These scenarios are designed to assess the adaptability and effectiveness of our clustering approach under varied data distribution conditions among devices.

- **Symmetric Scenario**: Devices either share identical data distributions or possess completely contrasting distributions. The formation of clusters is such that devices within the same cluster either mutually benefit or detrimentally affect each other's model training outcomes. Optimal performance is achieved when devices with similar distributions are clustered together, and those with divergent distributions are separated.

- **Asymmetric Scenario**: This scenario addresses the complexity arising from the diversity in device data distributions. It explores situations where devices are not reciprocal.

This section is structured as follows: We start by describing the baselines and datasets in Section 3.3.1, explore symmetric and asymmetric settings in Section 3.3.2, 3.3.3. Finally, we synthesize our findings and discuss their implications in Section 3.3.4.

3.3.1 Benchmarks

In our empirical evaluation, we compare our algorithms against well-established methods in Federated Learning:

- **Centralized Learning**: A baseline where all data is aggregated and processed in a central location.
- **FedAvg** [12]: A cornerstone algorithm in federated learning that averages locally updated models on the server.
- **FedGroup** [5]: Clusters devices based on the similarity of their updates and then performs federated learning within each cluster.
- **IFCA** [7]: Introduces cluster-based federated learning where each cluster receives a model tailored to its specific characteristics.
- **FeSEM** [19]: Utilizes a federated version of the stochastic expectation maximization algorithm to better handle non-IID data across clients.
- **FedDrift** [8]: Adapts to changes in data distribution over time, addressing the challenge of data drift in federated settings.
- **FedSoft** [14]: Operates similarly to IFCA but allows devices to participate in multiple clusters by determining a cluster importance score based on local data losses. This method is particularly relevant for comparison due to its flexible clustering mechanism.

We use the following datasets for our experiments:

- **MNIST** [4] A benchmark dataset comprising 28×28 pixel gray-scale images of handwritten digits, categorized into 10 classes.

- **Extended MNIST (EMNIST)** [3] An extension of MNIST to handwritten characters, offering a 62-class image classification challenge. For our experiments, we focus on the first ten characters from 'a' to 'j'.
- **Fashion MNIST (FASHION)** [18] Similar in structure to MNIST, this dataset features 28×28 pixel gray-scale images of fashion items, divided into 10 categories.
- **Federated Extended MNIST (FEMNIST)** [1] A federated learning-specific version of EMNIST, where each device's data originates from a unique writer, featuring a total of 3550 users. We utilize a subset comprising 5% of the data from 197 users.
- **CIFAR10** [10]: A popular benchmark dataset consisting of 32×32 pixel color images, divided into 10 classes, each representing different objects such as animals and vehicles.
- **Sentiment140 (SENT140)** [1]: A federated version of Text Dataset of Tweets. It contains 1,600,000 tweets extracted using Twitter.

3.3.2 Experiment Settings - Symmetric

In the symmetric setting, our objective is to evaluate the adaptability and effectiveness of our proposed federated learning technique under conditions of either homogeneous or extremely heterogeneous data distributions among devices. We conduct these evaluations using both natural and synthetic data partitions to simulate various distribution scenarios.

- **Natural Data Partition ($\mathbf{S_0}$)**: This setup simulates a scenario where each device's data is independently and identically distributed. This setup tests the algorithm's ability to recognize and maintain uniformity across devices in a federated environment.
- **Synthetic Label Partition ($\mathbf{S_1}$)**: In this setup, the dataset is allocated among devices based on distinct label ranges. For example, one set of devices might exclusively receive data corresponding to labels 0-4, while another set of devices receives data with labels 5-9. In this scenario, since devices with different label distributions are less likely to benefit each other, forming separate clusters for each label distribution is considered optimal.
- **Synthetic Predictor Partition ($\mathbf{S_2}$)**: In this partition, datasets of devices from different distributions have different underlying predictors. For instance, in the MNIST dataset, one group of devices (set 1) may map images directly to their corresponding

labels, while another group (set 2) maps images to shifted labels (e.g., mapping the image of digit 0 to label 1). This setup challenges the algorithm's ability to handle scenarios with significant variations in data mappings across devices. In such cases, combining data from different distributions can be harmful, indicating the need for distinct clusters to maintain the integrity of each device's predictive model.

3.3.3 Experiment Settings - Asymmetric

In the asymmetric setting, we explore a more complex scenario where devices' data distributions are not entirely contradictory, and some devices may benefit from collaborating with others from different distributions. This setting is designed to showcase the adaptability and performance advantages of our proposed federated learning technique in creating asymmetric clusters. We design experiments with both natural and synthetic partitions with respect to data quality and quantity.

- **Natural Data Partition ($\mathbf{A_0}$)**: Utilizing the FEMNIST subset of the LEAF dataset, we simulate a realistic asymmetric scenario. For instance, in the FEMNIST dataset, each device corresponds to a unique writer, allowing us to simulate a realistic asymmetric scenario. Given that different writers produce images of varying quality for classification, asymmetric clustering becomes beneficial. In such a setup, devices associated with high-quality writers can lend support to those linked to lower-quality writers, demonstrating the utility of asymmetric clusters in enhancing overall performance.
- **Synthetic Image Partition ($\mathbf{A_1}$)**: In this setup, one group of devices receives data with pristine quality (no noise), while another group handles data contaminated with Gaussian [16] and salt & pepper noise [2]. This partition serves to illustrate the need for asymmetric clustering, where devices with noisy data can benefit from the cleaner inputs of other devices. It tests the algorithm's ability to optimize learning outcomes in the presence of varying data quality.
- **Synthetic Data Amount Partition ($\mathbf{A_2}$)**: We construct a scenario where one set of devices has access to abundant data, while another set is limited in data quantity and exhibits slightly different predictors. Asymmetric clustering plays a crucial role in such environments. Devices with enough data, although reluctant to merge due to predictor discrepancies, can still offer valuable insights to devices with sparse data. Conversely, devices with limited data can leverage the more extensive datasets of others to reduce variance, even at the risk of introducing some bias.

3.3.4 Empirical Results

The outcomes of our experiments are presented across Tables 3.1 to 3.6. In both symmetric and asymmetric settings, we conducted experiments under both natural and synthetic data distribution scenarios. For the baseline setups, we begin with an initial allocation of five clusters, which is chosen based on the anticipated diversity within each dataset's data distributions. We set this figure as an upper boundary for the potential number of clusters, ensuring that the model has the capacity to accurately represent all possible clusters. This setup mirrors conditions in real-world federated learning scenarios, where the exact number of natural data clusters is unknown. By initializing more clusters than may be necessary, the baseline algorithms retain the flexibility to achieve accurate clustering by potentially leaving some clusters empty. The table below presents the communication costs observed during the MNIST experiment, which incorporated 20 devices per iteration. This table compares the communication overhead associated with our approach relative to conventional federated learning methods. Although our method initially results in increased communication demands in the first $\log n$ iterations due to forming clusters, the overhead in subsequent iterations reduces to levels comparable to other baselines.

Table 3.1: Test accuracies ± stderr with [number of clusters] in S_0.

Dataset	MNIST	EMNIST	FASHION	CIFAR10
Centralize	97.64±0.02 [1]	98.08±0.05 [1]	89.21±0.07 [1]	76.42±0.06 [1]
FedAvg	**96.24±0.10** [1]	**97.71±0.08** [1]	**88.63±0.21** [1]	**73.28±0.13** [1]
IFCA	95.05±0.23 [3]	94.54±0.83 [1]	85.67±0.38 [4]	71.46±0.37 [5]
FeSEM	94.29±0.44 [3]	94.20±1.13 [1]	86.33±0.18 [1]	68.43±0.16 [3]
FedGroup	95.78±0.23 [5]	96.09±0.20 [5]	86.19±0.08 [5]	72.27±0.14 [5]
FedDrift	95.66±0.82 [2]	97.07±0.26 [3]	86.98±0.67 [3]	71.47±0.50 [2]
FedSoft	96.03±0.21 [5]	93.21±0.19 [5]	83.58±0.22 [5]	72.74±0.18 [5]
FAACL(ours)	96.12±0.99 [1]	97.09±0.28 [1]	88.24±0.44 [1]	72.53±0.23 [1]

- **Symmetric Natural Distribution (S_0)**: In scenarios with natural data distributions, our method, FAACL, demonstrates performance that is comparable to FedAvg. FAACL is not expected to outperform FedAvg in this setting, as these conditions are precisely those for which FedAvg is optimized. It is important to note that other clustered federated learning methods, including IFCA, FeSEM, FedSoft, and FedGroup,

Table 3.2: Test accuracies ± stderr with [number of clusters] in A_0.

Dataset	FEMNIST	Sent140
Centralize	74.26±0.14 [1]	79.26±0.17 [1]
FedAvg	61.98±0.39 [1]	70.49±0.08 [1]
IFCA	51.21±0.38 [5]	72.86±0.24 [5]
FeSEM	47.52±3.92 [1]	63.69±1.83 [2]
FedGroup	65.47±1.02 [5]	72.38±0.43 [5]
FedDrift	62.28±0.39 [8]	73.86±0.68 [14]
FedSoft	67.87±0.91 [5]	72.26±1.12 [5]
FAACL(ours)	**69.33±0.05** [13]	**75.17±0.71** [11]

initiate with a fixed number of clusters. This can lead to under-trained models due to data dilution across too many clusters, especially when the fixed number does not align with the optimal cluster count for the given data distribution. Although FedDrift is capable of dynamically determining the number of clusters during training, it still tends to output more clusters than required.

- **Symmetric Synthetic Distribution ($\mathbf{S_1}, \mathbf{S_2}$)**: In the synthetic distribution (i.e., synthetic labels and synthetic predictors), our method still outperforms other baselines in most datasets due to its adaptive number of clusters. FedAvg struggles in the synthetic scenarios because it always groups all devices into one cluster. Specifically, in the synthetic predictor setting where devices have conflicting predictors, FedAvg struggles significantly because it groups all devices into one cluster, leading to about 50% accuracy, which highlights the limitation of keeping all devices in one cluster. Our findings indicate that while FAACL specializes in generating effective asymmetric clusters, it does so without sacrificing performance when compared to methods that typically learn symmetric clusters, thereby showcasing its robust adaptability and efficiency in diverse settings.

- **Asymmetric Natural Distribution ($\mathbf{A_0}$)**: In the natural distribution, the FEMNIST and Sent140 dataset from the LEAF benchmark are used to construct each device to represent a writer / user. In this natural distribution, the true number of clusters is unknown, we train the centralized method with all devices in one cluster. In this experiment with real data, our methods demonstrated high accuracies compared

Table 3.3: Test accuracies ± stderr for S_1 with [number of clusters].

Dataset	**MNIST**	**EMNIST**	**FASHION**	**CIFAR10**
Centralize	94.85±0.16[2]	97.11±0.09[2]	90.82±0.12[2]	74.68±0.11[2]
FedAvg	81.86±0.79[1]	90.21±0.49[1]	82.89±0.28[1]	51.85±0.37[1]
IFCA	91.16±0.38[5]	94.28±0.46[4]	86.88±0.24[4]	69.36±0.31[5]
FeSEM	50.24±3.80[3]	42.44±3.88[1]	50.57±1.55[1]	54.20±0.71[1]
FedGroup	**93.73±0.13**[5]	96.18±0.14[5]	88.50±0.44[5]	**71.62±0.29**[5]
FedDrift	91.76±0.11[8]	96.35±0.20[3]	85.52±0.24[7]	70.53±0.79[3]
FedSoft	90.49±0.25 [5]	94.39±0.71 [5]	84.29±0.36 [5]	72.49±0.33[5]
FAACL(ours)	93.44±0.05[2]	**96.86±0.11**[2]	**90.22±0.18**[2]	71.40±0.36[2]

Table 3.4: Test accuracies ± stderr for S_2 with [number of clusters].

Dataset	**MNIST**	**EMNIST**	**FASHION**	**CIFAR10**
Centralize	97.06±0.04[2]	96.93±0.13[2]	88.95±0.68[2]	73.57±0.11[2]
FedAvg	51.36±0.99[1]	47.30±0.87[1]	44.42±1.14[1]	67.96±0.52[1]
IFCA	94.36±0.52[4]	95.17±0.05[2]	85.42±0.48[5]	71.11±0.23[4]
FeSEM	49.35±4.17[3]	43.94±3.05[1]	43.65±1.65[1]	64.76±1.32[3]
FedGroup	95.55±0.22[5]	95.79±0.22[5]	85.98±0.09[5]	70.81±0.41[5]
FedDrift	93.37±0.35[6]	96.12±0.17[3]	85.77±0.28[4]	71.30±0.55[3]
FedSoft	93.92±0.41 [5]	94.78±0.63 [5]	85.11±0.26 [5]	**71.58±0.57**[5]
FAACL(ours)	**95.73±0.02**[2]	**96.48±0.02**[2]	**87.24±0.08**[2]	71.46±0.37[2]

to other clustered baselines, including FedDrift which can form adaptive clusters by at least 5%. This improvement in the real-world dataset underscores the advantage of asymmetric clustering.

- **Asymmetric Synthetic Distribution ($\mathbf{A_1}, \mathbf{A_2}$)**: In the synthetic experiments, we note that performance between FedAvg and other clustered methods varies across settings (i.e., FedAvg may outperform other clustered based methods in some dataset, while in others, clustered based methods achieve higher accuracy.). This variability is influenced by the trade-off between the increased accuracy gained by one device and

Table 3.5: Test accuracies ± stderr for A_1 with [number of clusters].

Dataset	**MNIST**	**EMNIST**	**FASHION**	**CIFAR10**
Centralize	78.39±0.05[2]	62.39±0.32[2]	73.43±0.13[2]	58.51±0.04[2]
Fedavg	70.33±0.29[1]	51.00±0.20[1]	72.99±0.31[1]	52.91±0.41[1]
IFCA	69.90±0.41[5]	50.50±0.84[5]	69.09±0.61[4]	53.29±0.51[5]
FeSEM	65.79±0.82[4]	46.31±2.24[1]	64.56±1.32[1]	48.72±0.92[1]
FedGroup	74.65±0.16[5]	51.88±0.24[5]	70.93±0.52[5]	54.26±0.58[5]
FedDrift	70.34±0.43[7]	51.63±0.11[4]	72.96±0.17[3]	54.58±0.46[1]
FedSoft	71.28±0.31 [5]	52.38±0.56[5]	71.01±0.47 [5]	53.38±0.75[5]
FAACL(ours)	**75.12±0.23**[4]	**53.08±0.33**[9]	**73.19±0.06**[3]	**55.72±0.57**[6]

Table 3.6: Test accuracies ± stderr for A_2 with [number of clusters].

Dataset	**MNIST**	**EMNIST**	**FASHION**	**CIFAR10**
Centralize	97.32±0.03[2]	97.67±0.29[2]	86.90±0.24[2]	64.16±0.09[2]
Fedavg	89.77±0.31[1]	89.30±0.17[1]	83.00±0.01[1]	58.48±0.23[1]
IFCA	95.35±0.28[3]	95.98±0.18[2]	85.70±0.32[4]	59.17±0.62[5]
FeSEM	74.65±0.20[1]	88.02±1.27[1]	81.27±0.30[1]	52.84±1.28[5]
FedGroup	95.44±0.26[5]	95.42±0.18[5]	86.31±0.28[5]	60.27±0.61[5]
FedDrift	95.34±0.16[3]	91.70±0.45[4]	84.15±0.35[2]	60.28±1.14[8]
FedSoft	95.73±0.09 [5]	95.73±0.09 [5]	84.29±0.58 [5]	60.25±0.79[5]
FAACL(ours)	**96.01±0.52**[5]	**96.95±0.18**[4]	**86.52±0.01**[2]	**62.39±0.61**[8]

the potential decrease in accuracy for the other devices in different distributions. In contrast, our approach to constructing asymmetric clustering consistently maintains a cluster for each device for prediction purposes. Then each device can contribute to the training of other clusters without worrying about hurting its performance.

The Table 3.7 below presents the communication costs observed during the MNIST experiment, which incorporated 20 devices per iteration. This table compares the communication overhead associated with our approach relative to conventional federated learning methods. Although our method initially results in increased communication demands in the

first $\log n$ iterations due to forming clusters, the overhead in subsequent iterations reduces to levels comparable to other baselines.

Table 3.7: Communication Overhead (combined size of all messages between the server and the devices in one communication round) in MNIST Experiments

	S_0	S_1	S_2	A_1	A_2
FedAvg	16.3 MB	16.3 MB	16.3 MB	16.3 MB	16.3 MB
IFCA	48.9 MB	48.9 MB	48.9 MB	48.9 MB	48.9 MB
FeSEM	48.9 MB	48.9 MB	48.9 MB	48.9 MB	48.9 MB
FedGroup	16.3 MB	16.3 MB	16.3 MB	16.3 MB	16.3 MB
FedDrift	24.5 MB	73.4 MB	57.1 MB	65.2 MB	32.6 MB
FedSoft	81.5 MB	81.5 MB	81.5 MB	81.5 MB	81.5 MB
FAACL(first $\log 20 \approx 5$ rounds)	70.1 MB	70.1 MB	70.1 MB	101.1 MB	101.1 MB
FAACL(after $\log 20 \approx 5$ rounds)	16.3 MB	16.3 MB	16.3 MB	20.4 MB	20.4 MB

A challenge in the asymmetric setting is determining the threshold between different distributions. Given the divergence in data distributions and the varying impact of clustering on different devices, finding the ideal number of clusters can be non-trivial. However, the experiments show that our method's capability to perform asymmetric clustering, while not necessarily finding the number of correct clusters, consistently delivers the best accuracies in various settings.

Chapter 4

Continual Federated Adaptive Asymmetric Clustered Learning

In the continual learning extension of our federated clustering framework, we enhance the framework to address both spatial and temporal heterogeneity across different time-steps.

4.1 Problem Statement

Building on the loss function defined in Equation 3.1, our objective in the continual learning setting is to optimize the cumulative performance of the model across all time-steps. Specifically, we aim to minimize the total loss across the series of time-steps, expressed mathematically as:

$$\sum_{i=1}^{T} L(\mathcal{C}^{(t)})$$

Here, $L(\mathcal{C}^{(t)})$ is the loss associated with the clustering $\mathcal{C}^{(t)}$ at each time-step t. The clustering $\mathcal{C}^{(t)}$ is dynamically updated at each time step to incorporate the latest data.

The ultimate goal of the continual learning extension is not just to adapt to new data but to optimize the overall system's performance over time. This involves balancing the need for adaptation against the need to maintain stability and prevent drastic shifts in the model that could lead to increased errors or loss of valuable insights.

4.2 Algorithm Details

4.2.1 Cold Start Phase ($t = 0$)

In the continual learning extension, the initial clustering cold start phase utilizes the original FAACL framework. This phase begins with each device as a separate cluster and gradually refines these clusters to establish a base clustering for subsequent time steps, as detailed in Algorithm 7.

Algorithm 7 Cold Start

Input: Device set $\mathcal{D}$, significance level α, threshold ϵ, number of epochs *epochs*
Output: Clustering $\mathcal{C}$

$\mathcal{C} = FAACL(\mathcal{D}, \alpha, \epsilon, epochs)$
for each device d in $\mathcal{D}$ **do**
  Update device memory structure for the initial time step $d.\mathcal{M}^{(0)}$ with:
    - Cluster membership
    - Local samples
for each cluster C in $\mathcal{C}$ **do**
  Update cluster memory structure for initial time step $C.\mathcal{M}^{(0)}$ with all assigned devices
Return $\mathcal{C}$

4.2.2 Local Concept Search

The Local Concept Search is designed to handle concept drift by utilizing historical clustering information stored in a replay buffer. This process helps each device autonomously determine if the new observed data matches a previously encountered concept. The algorithm 8 demonstrate the steps with explanation as below:

1. **Data Reception and Model Training**: Each device d receives new data $Z_d^{(t)}$. A new local model, $d.\theta^{(t)}$ is then trained to fit the new dataset.

2. **Concept Recurrence Detection**: The device evaluates potential concept drift by comparing the new model $d.\theta^{(t)}$ against previous models $\{d.\theta^{(t')} : t' < t\}$ stored in the replay buffer. Both $d.\theta^{(t)}$ and $d.\theta^{(t')}$ are tested on the current validation set $Z_d^{val_{(t)}}$ and previous data samples to generate validation losses.

3. **Cluster Assignment**: Statistical tests (e.g. Wilcoxon signed-rank test) are applied to the validation losses to evaluate the performance between both models. If the test results indicate similar model performance on both datasets, this suggests that the concept reoccurs. Consequently, the device is assigned to the same cluster it was assigned to at time t'.

By employing the local concept search approach, each device autonomously determines whether its current concept has been previously encountered. If so, the device can be directly assigned to the appropriate cluster managed similar concepts.

Algorithm 8 Local Concept Search

Input: Device d, Clustering $\mathcal{C}_{t-1}$, significance level α
Output: Boolean value indicating whether concept has been found.

Train new model $d.\theta$ on the new data Z_d^{train}.
for data sample, cluster $\{S', C'\}$ in $\{d.\mathcal{M}^{(t')} \mid t' < t\}$ **do**
  $l_1 = loss(C'.\theta, Z_d^{val})$, $l_2 = loss(d.\theta, Z_d^{val})$ $l_3 = loss(C'.\theta, S')$, $l_4 = loss(d.\theta, S')$
  $p_1 = Wilcoxon(l_1, l_2)$, $p_2 = Wilcoxon(l_3, l_4)$
  if $p_1 < \alpha$ **and** $p_2 < \alpha$ **then**
    Assign d to C'
    Return True.
Return False.

4.2.3 Global Concept Search

In case where the Local Concept Search does not yield a match between new data and previously identified concepts in a device's replay buffer, the Global Concept Search extends the search scope to include clusters from the previous clustering $\mathcal{C}_{t-1}$. The steps are shown in Algorithm 9 with details below:

1. **Cluster Model Comparison**: Each device d compares its new model $d.\theta^{(t)}$ against all cluster models in $\mathcal{C}_{t-1}$ on the local data $Z_d^{val_{(t)}}$, and on the data from previous devices within the cluster from the last time step.

2. **Concept Recurrence Detection**: A statistical test (Wilcoxon signed-rank test) is conducted to compare the validation losses. If the test results show that both models perform similarly, the local device is then assigned to this cluster.

The Global Concept Search aims to efficiently integrate new device data into the existing clustered framework by reusing and adapting existing clusters whenever possible. This approach minimizes the need to create new clusters unnecessarily, promoting stability and reducing computational overhead within the federated learning system.

Algorithm 9 Global Concept Search

Input: Device d, Device set $\mathcal{D}$, significance level α, Clustering $\mathcal{C}_{t-1}$, current timestep t
Output: Boolean value indicating whether concept has been found

---

for all cluster C in $\mathcal{C}_{t-1}$ **do**
  Query from $\{C.\mathcal{M}^{(t')} \mid t' < t\}$ to select d' at t' with $d'.\mathcal{M}^{(t')} = \{S', C\}$
  $l_1 = loss(C.\theta, Z_d^{val})$, $l_2 = loss(d.\theta, Z_d^{val})$ $l_3 = loss(C.\theta, S')$, $l_4 = loss(d.\theta, S')$
  $p_1 = Wilcoxon(l_1, l_2)$, $p_2 = Wilcoxon(l_3, l_4)$
  if $p_1 < \alpha$ **and** $p_2 < \alpha$ **then**
    Assign cluster C to device d
    Update $d.\mathcal{M}^{(t)} = \{S_{val}, C\}$ where S_{val} sampled from Z_d^{val}
    Return $True$
Return $False$

4.2.4 Establish New concept

In instances where both Local and Global Concept Searches fail to identify any matching clusters for the newly observed data, this suggests that the device has encountered a new concept previously unrepresented in the system. To accommodate this new concept, a fresh cluster is initiated and integrated into the existing clustering framework. The steps are shown in Algorithm 10 with details below:

1. **Initialization of a new cluster** A new cluster is created using the local model as the cluster model. Initially, this cluster contains only the local device.
2. **Interaction with other clusters**: The newly formed cluster then engages in model exchanges with existing clusters to explore potential supportive relationships. These relationships are determined based on the validation losses incurred when other clusters evaluate the new model on their respective datasets.
3. **Integration to the system**: Once supportive relationships are established, the new cluster is formally added to the clustering configuration $\mathcal{C}_t$.

Algorithm 10 New Concept

Input: Device d, Clustering $\mathcal{C}_{t-1}$
Output: Clustering $\mathcal{C}$

Train new model $d.\theta$ on the new data Z_d^{train}.
Establish a new cluster C with $C.D = \{d\}$ and model $C.\theta = d.\theta$
$\mathcal{C}.add(C)$
Update the $d.\mathcal{M}^{(t)} = \{S_{val}, C\}$
Return clustering $\mathcal{C}$

4.2.5 Proposed Continual Extension

The approach outlined in this section integrates the various components discussed earlier into an overall method specifically designed to address the challenges of continual learning within a federated learning framework. The process initiates with a 'cold start' phase, employing the standard FAACL procedures. Subsequently, in the next time step, the system tries to assign devices to clusters by identifying concepts within both local and global ranges. If no existing concept matches the new data, the system proceeds to establish a new concept. This process is detailed in Algorithm 11.

Algorithm 11 C-FAACL

Input: Device set $\mathcal{D}$, significance level α, threshold ϵ, number of epochs *epochs*, total time step T
Output: Clustering $\mathcal{C}_t$

Clustering $\mathcal{C} = ColdStart(\mathcal{D}, \alpha, \epsilon, epochs)$
for time t from 1 to T **do**
  for device d in $\mathcal{D}$ **do**
    if *Local Concept Search*$(d, \alpha, \mathcal{C}_{t-1})$ **then**
      Continue
    else if *Global Concept Search*$(d, \alpha, \mathcal{C}_{t-1}, t)$ **then**
      Continue
    else
      $\mathcal{C} = New\ Concept(d, \mathcal{C}_{t-1})$
  Cluster Train$(\mathcal{C})$
  Cluster Merge$(\mathcal{C})$
Return Clustering $\mathcal{C}$.

4.3 Experiments

In this section, we present results for the continual learning extension, which involves non-iid concepts in both symmetric and asymmetric cases. The experiments are designed under two distinct settings: Sequential Task Learning and Multi-Task Local Learning. These settings are intended to assess the model's adaptability to different types of data heterogeneity and its ability to manage learning challenges over time.

- **Sequential Task Learning**: This setup is focused on temporal heterogeneity, where data concepts are introduced sequentially over time. The main objective is to evaluate how effectively the model adapts to new tasks while retaining proficiency in previously learned tasks, thus addressing the challenge of catastrophic forgetting.
- **Multi-Task Learning**: : This scenario contrasts with the sequential approach by incorporating both temporal and spatial heterogeneity. Here, multiple tasks are introduced simultaneously, challenging the model's capacity to handle complex data distributions and to maintain performance across concurrent learning challenges.

The experiments involve two non-iid concepts extended over 10 timesteps, with each timestep consisting of 100 epochs. We compare the proposed techniques with Federated Continual learning baselines, including FedProx-EWC, FedProx-APD, FedProx-SSGD [11], FedDrift [8], and FedWeIT [21].

This section is structured as follows: We explore Sequential Learning and Multi-task Learning in Section 4.3.1 and Section 4.3.2. Finally we discuss the results and their implications in Section 4.3.3.

4.3.1 Experiment Setting - Sequential Task Learning

The primary goal of this experiment is to examine how the system adaptively manages new data over time without sacrificing the retention of previously learned information. This setup is designed to evaluate the framework's resilience to catastrophic forgetting, particularly in scenarios where new tasks are introduced sequentially over a series of time steps.

This configuration represents the simplest scenario in federated continual learning by assuming homogeneity among devices. Such homogeneity allows the server to aggregate updates from all devices without worrying about the conflict between updates. The focus here is primarily on addressing challenges associated with temporal heterogeneity. In this setup, all devices simultaneously receive the same tasks, ensuring that they share identical learning objectives throughout each phase of the experiment.

Table 4.1: Test accuracies for symmetric concepts S_1, S_2 in Sequential Task Learning.

Dataset	**MNIST**		**FASHION**		**CIFAR10**	
	S_1	S_2	S_1	S_2	S_1	S_2
FedProx-EWC	93.51 ± 0.09	90.26 ± 0.43	63.07 ± 0.39	72.00 ± 0.71	79.78 ± 0.48	87.59 ± 0.25
FedProx-APD	97.23 ± 0.06	86.59 ± 0.07	68.44 ± 0.21	73.65 ± 0.49	78.17 ± 0.57	85.02 ± 0.48
FedProx-SSGD	96.42 ± 0.26	89.20 ± 0.35	61.68 ± 0.24	72.75 ± 0.15	81.59 ± 0.30	85.84 ± 0.54
FedDrift	96.55 ± 0.40	$\mathbf{97.83} \pm 0.22$	$\mathbf{75.80} \pm 0.28$	$\mathbf{74.95} \pm 0.64$	$\mathbf{93.06} \pm 0.49$	87.62 ± 0.28
FedWeIT	$\mathbf{97.81} \pm 0.33$	87.98 ± 0.07	66.49 ± 0.58	73.86 ± 0.22	82.57 ± 0.44	$\mathbf{89.51} \pm 0.45$
FAACL (ours)	97.00 ± 0.16	97.67 ± 0.25	75.54 ± 0.47	74.21 ± 0.10	92.28 ± 0.41	89.43 ± 0.31

For this experiment, the dataset is partitioned into four distinct types of concept settings to rigorously evaluate our method under diverse conditions. In the symmetric settings, characterized by S_1 where concepts differ in their label ranges and S_2 where they differ

Table 4.2: Test accuracies for asymmetric concepts A_1, A_2 in Sequential Task Learning.

Dataset	**MNIST**		**FASHION**		**CIFAR10**	
	A_1	A_2	A_1	A_2	A_1	A_2
FedProx-EWC	83.29 ± 0.22	81.14 ± 0.43	66.52 ± 0.41	65.54 ± 0.33	78.65 ± 0.23	82.31 ± 0.60
FedProx-APD	85.83 ± 0.49	83.65 ± 0.53	66.34 ± 0.19	66.19 ± 0.25	79.36 ± 0.62	83.06 ± 0.33
FedProx-SSGD	86.18 ± 0.46	85.61 ± 0.30	66.74 ± 0.39	67.05 ± 0.55	80.23 ± 0.38	82.99 ± 0.41
FedDrift	89.83 ± 0.28	89.26 ± 0.63	68.27 ± 0.59	69.95 ± 0.46	81.66 ± 0.43	84.53 ± 0.22
FedWeIT	88.62 ± 0.41	87.84 ± 0.51	70.74 ± 0.40	67.10 ± 0.66	80.96 ± 0.35	85.61 ± 0.43
FAACL (ours)	$\mathbf{91.97} \pm 0.08$	$\mathbf{90.96} \pm 0.14$	$\mathbf{71.38} \pm 0.08$	$\mathbf{71.16} \pm 0.48$	$\mathbf{82.19} \pm 0.31$	$\mathbf{86.37} \pm 0.23$

in their predictors, our objective was not to necessarily surpass existing techniques but to ensure that C-FAACL has competitive performance. The results from these settings demonstrate that C-FAACL is capable of achieving this goal, confirming its efficacy in environments where data distributions are uniform across devices.

In the asymmetric settings, which are more challenging due to variations in data quality as seen in A_1 (differences in image quality/noise) and variations in both data quantity and predictors as seen in A_2, C-FAACL particularly excels. These settings highlight the unique capability of C-FAACL to dynamically generate asymmetric clusters, a feature that traditional federated continual learning techniques often lack. In these scenarios, C-FAACL consistently outperforms other methods, effectively leveraging its novel clustering approach to produce asymmetric clustering. This underscores the adaptability of C-FAACL with its superior performance in handling complex, real-world data distributions compared to traditional Federated Continual Learning method. The outcomes of these experiments are shown in Tables 4.1, 4.2, and will be discussed in Section 4.3.3.

4.3.2 Experiment Setting - Multi-Task Learning

In this experiment, we explore the complexities of Multi-Task Learning where each device is engaged in potentially different tasks at the same time. This setup introduces a layer of complexity that evaluates the model's ability to manage task diversity effectively. The central aim is to assess the system's capability to learn shared representations that are beneficial across varying tasks.

Compared to Sequential Task Learning, Multi-Task Learning presents greater complexity as it encompasses both spatial and temporal heterogeneity. In this dynamic setting, not only do concepts evolve over time, but devices may also engage with different concepts

Table 4.3: Test accuracies for symmetric concepts S_1, S_2 in Multi-Task Learning.

Dataset	**MNIST**		**FASHION**		**CIFAR10**	
	S_1	S_2	S_1	S_2	S_1	S_2
FedProx-EWC	90.21 ± 0.26	49.15 ± 0.69	50.94 ± 0.30	68.28 ± 0.44	77.81 ± 0.23	84.18 ± 0.54
FedProx-APD	93.47 ± 0.52	80.48 ± 0.94	51.42 ± 0.44	70.19 ± 0.61	75.17 ± 0.16	84.33 ± 0.45
FedProx-SSGD	94.82 ± 0.35	49.98 ± 0.52	51.00 ± 0.42	67.71 ± 0.38	78.45 ± 0.65	82.89 ± 0.34
FedDrift	$\mathbf{95.94} \pm 0.11$	95.25 ± 0.69	$\mathbf{74.27} \pm 0.57$	71.39 ± 0.51	$\mathbf{90.64} \pm 0.23$	87.02 ± 0.70
FedWeIT	94.39 ± 0.05	84.34 ± 0.79	50.20 ± 0.36	69.91 ± 0.61	81.47 ± 0.78	84.33 ± 0.29
FAACL (ours)	94.73 ± 0.41	$\mathbf{96.41} \pm 0.46$	73.16 ± 0.55	$\mathbf{73.01} \pm 0.63$	89.84 ± 0.64	$\mathbf{87.90} \pm 0.48$

Table 4.4: Test accuracies for asymmetric concepts A_1, A_2 in Multi-Task Learning.

Dataset	**MNIST**		**FASHION**		**CIFAR10**	
	A_1	A_2	A_1	A_2	A_1	A_2
FedProx-EWC	81.11 ± 0.66	79.42 ± 0.22	63.30 ± 0.84	61.96 ± 0.37	75.15 ± 0.68	79.62 ± 0.75
FedProx-APD	81.69 ± 0.76	81.81 ± 0.62	60.99 ± 0.25	62.43 ± 0.14	76.02 ± 0.42	78.33 ± 0.77
FedProx-SSGD	82.80 ± 0.21	83.16 ± 0.22	63.15 ± 0.78	63.81 ± 0.43	75.86 ± 0.30	78.09 ± 0.73
FedDrift	84.49 ± 0.46	85.58 ± 0.74	64.51 ± 0.18	64.16 ± 0.39	79.32 ± 0.28	81.05 ± 0.37
FedWeIT	85.16 ± 0.75	83.64 ± 0.82	64.94 ± 0.48	64.28 ± 0.37	77.74 ± 0.39	80.23 ± 0.25
FAACL (ours)	$\mathbf{88.15} \pm 0.30$	$\mathbf{87.06} \pm 0.73$	$\mathbf{68.62} \pm 0.89$	$\mathbf{69.06} \pm 0.48$	$\mathbf{80.86} \pm 0.59$	$\mathbf{83.01} \pm 0.71$

at any given time step. This scenario thoroughly tests the model's adaptability and its capacity to handle concurrent learning challenges effectively.

To facilitate a direct comparison with the Sequential Task Learning experiment, we employ the same four distinct types of concept settings: (S_1, S_2, A_1, A_2) same as the Sequential Task Learning experiment. This consistency ensures that the differences in model performance can be attributed to the learning paradigm rather than variations in the data concepts. The result of this setup is provided in Tables 4.3, 4.4.

4.3.3 Empirical Results

The results of our experiments are detailed from Table 4.1 to Table 4.4.

- **Asymmetric Settings Performance**: Amongst the asymmetric experiments, C-FAACL consistently outperforms the baseline methods. This superior performance

highlights the effectiveness of C-FAACL's one-way communication design in managing asymmetric data distributions, thereby enhancing overall clustering performance.

- **Multi-task vs. Sequential Task Performance**: A notable disparity in performance between ensemble methods (such as C-FAACL and FedDrift) and non-ensemble methods is observed when comparing Multi-Task Learning with Sequential Task Learning. This contrast is particularly stark in challenging scenarios such as S_2, where devices are presented with conflicting predictors. C-FAACL and FedDrift excel in these extreme conditions by dynamically assigning devices to appropriate clusters based on differing concepts, whereas non-ensemble methods, which attempt to use a single model to predict on contradictory data, tend to perform poorly. This underlines the importance of ensemble techniques in environments characterized by high heterogeneity.

Overall, the C-FAACL framework demonstrates high performance across a variety of experimental settings, effectively managing both temporal and spatial data heterogeneity. Its adaptability to different learning scenarios underscores its potential as a robust tool for federated learning.

Chapter 5

Conclusion

In this study, we introduced the innovative concept of asymmetric clustering to tackle specific challenges in federated learning, where the benefits of clustering are unevenly distributed among devices. Some devices significantly benefit from joining a new cluster, while others may experience detrimental effects due to diverse data characteristics. To address these challenges, we developed two key methodologies:

- **Federated Adaptive Asymmetric Clustering Learning (FAACL):** Designed to facilitate the formation of asymmetric clusters, optimizing the collaborative potential among devices with varying data distributions.
- **Continual Federated Adaptive Asymmetric Clustering Learning (C-FAACL):** An extension of FAACL that integrates continual learning capabilities, allowing for dynamic adaptation over time in response to evolving data distributions.

Our empirical evaluations demonstrated FAACL's superior performance compared to traditional symmetric clustering approaches, particularly when applied to real-world datasets. We adopted a partition-based strategy that effectively reduced the computational complexity to $O(n^2)$ for the entire process within $O(\log n)$ iterations. The continual learning extension, C-FAACL, showcased robustness in environments characterized by both temporal and spatial data heterogeneity. Through sequential task learning and multi-task learning experiments, C-FAACL efficiently managed new data over time, maintaining the integrity of previously learned information.

5.1 Limitation

Despite the promising results, our study has several limitations that should be addressed in future research:

- **Generalizability**: The current frameworks require all devices to use the same model architecture for training, which may not be feasible in heterogeneous network environments where devices have varying computational capabilities.
- **Complexity**: Compared to methods with a fixed number of clusters, our approach requires more communication and increased run-time complexity, which could hinder scalability.
- **Sensitive hyperparameter choice**: The selection of hyperparameters in FAACL significantly affects training convergence time and the resulting clustering configuration, necessitating careful tuning and potential automation.

5.2 Future Work

To build on the findings of this study, future research could explore:

- **Stable adaptive clustering**: The current FAACL model demonstrates a sensitivity to hyperparameter settings, necessitating careful tuning to achieve optimal performance. Future efforts could focus on developing adaptive mechanisms within the algorithm itself. These mechanisms would automatically adjust hyperparameters in response to dynamic changes in data characteristics, thereby reducing the model's sensitivity to initial settings and enhancing its robustness.
- **Broader Generalizability**: Expanding the applicability of the framework to accommodate devices with varying model architectures is another promising direction. This enhancement could involve integrating model-agnostic federated learning techniques, which would enable a more inclusive and flexible training environment, catering to a diverse array of devices and data types.

References

[1] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

[2] Kenneth R Castleman. *Digital image processing*. Prentice Hall Press, 1996.

[3] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*, pages 2921–2926. IEEE, 2017.

[4] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142, 2012.

[5] Moming Duan, Duo Liu, Xinyuan Ji, Renping Liu, Liang Liang, Xianzhang Chen, and Yujuan Tan. Fedgroup: Efficient federated learning via decomposed similarity-based clustering. In *2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, pages 228–237. IEEE, 2021.

[6] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning: A meta-learning approach. *arXiv preprint arXiv:2002.07948*, 2020.

[7] 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.

[8] Ellango Jothimurugesan, Kevin Hsieh, Jianyu Wang, Gauri Joshi, and Phillip B Gibbons. Federated learning under distributed concept drift. In *International Conference on Artificial Intelligence and Statistics*, pages 5834–5853. PMLR, 2023.

[9] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[10] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[11] 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.

[12] 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.

[13] Bernard Rosner, Robert J Glynn, and Mei-Ling T Lee. The wilcoxon signed rank test for paired comparisons of clustered data. *Biometrics*, 62(1):185–192, 2006.

[14] Yichen Ruan and Carlee Joe-Wong. Fedsoft: Soft clustered federated learning with proximal local updating. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8124–8131, 2022.

[15] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE transactions on neural networks and learning systems*, 32(8):3710–3722, 2020.

[16] Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[17] Robert F Woolson. Wilcoxon signed-rank test. *Wiley encyclopedia of clinical trials*, pages 1–3, 2007.

[18] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[19] Ming Xie, Guodong Long, Tao Shen, Tianyi Zhou, Xianzhi Wang, Jing Jiang, and Chengqi Zhang. Multi-center federated learning. *arXiv preprint arXiv:2108.08647*, 2021.

[20] Xin Yang, Hao Yu, Xin Gao, Hao Wang, Junbo Zhang, and Tianrui Li. Federated continual learning via knowledge fusion: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[21] Jaehong Yoon, Wonyong Jeong, Giwoong Lee, Eunho Yang, and Sung Ju Hwang. Federated continual learning with weighted inter-client transfer. In *International Conference on Machine Learning*, pages 12073–12086. PMLR, 2021.

[22] Jaehong Yoon, Saehoon Kim, Eunho Yang, and Sung Ju Hwang. Scalable and order-robust continual learning with additive parameter decomposition. *arXiv preprint arXiv:1902.09432*, 2019.

[23] Jerrold H Zar. *Biostatistical analysis*. Pearson Education India, 1999.

APPENDICES

Appendix A

Algorithm Analysis

The complexity of each clustering algorithm within our framework is analyzed to understand the computational demands of the process.

Proposition 1. *The Cluster Initialization algorithm operates with a complexity of* $O(n)$.

Proof. Creating one cluster per device for n devices directly results in a complexity of $O(n)$, as detailed in Algorithm 1. □

Proposition 2. *The Cluster Support algorithm operates with a complexity of* $O(n^2)$.

Proof. With $m{=}O(n)$ clusters, each cluster's potential supportive clusters are assessed among all others, resulting in a complexity bounded by the product of the number of clusters $m{=}O(n)$ and the maximum number of devices per cluster $O(n)$, yielding $O(n^2)$. □

Proposition 3. *The Cluster Merge algorithm operates with a complexity of* $O(n^2)$.

Proof. With $m{=}O(n)$ clusters, ths algorithm goes through each pair of cluster, resulting in a complexity $O(m^2){=}O(n^2)$. □

Proposition 4. *The Cluster Training algorithm operates with a complexity of* $O(n^2)$.

Proof. Given $m{=}O(n)$ clusters, the training process for a cluster model involves all devices in the cluster, along with all devices from its supportive clusters. For a cluster C, the number of devices participating its training process is at most $|C.D|+\sum_{C'\in C.sup}|C'.D|$, thus the overall complexity is $\sum_{i=1}^{m}(|C_i.D|+\sum_{C'\in C_i.sup}|C'.D|){=}\sum_{i=1}^{m}|C_i.D|+\sum_{i=1}^{m}\sum_{C'\in C_i.sup}|C'.D|{=}n+mn{=}O(n^2)$. □

Proposition 5. *The Flat FAACL algorithm operates with a per-iteration complexity of $O(n^2)$. When provided with an initial clustering of size $O(1)$, the total computational complexity remains $O(n^2)$. When not provided with initial clustering, the total computational complexity is $O(n^3)$.*

Proof. Starting with an initialization phase of $O(n)$ complexity, Flat FAACL involves training and merging phases within each iteration, both of which contribute to a per-iteration complexity of $O(n^2)$ from the Proposition 1, 2, 3, 4. Given that the algorithm's convergence criteria are met within a finite number of iterations, and assuming the initial clustering involves a minimal number of clusters ($O(1)$), Flat FAACL effectively operates with an overall complexity of $O(n^2)$. This is due to the fact that the number of iterations required for convergence does not significantly alter the computational load, which is dominated by the costs of training and merging operations within each iteration. When the initial clustering is not provided, as the initial clustering has size of $O(n)$, the total iteration is $O(n)$, therefore the total complexity is $O(n^3)$. □

Theorem 6. *Hierarchical FAACL reduces the overall complexity to $O(n^2)$, in $\log n$ iterations of $O(n^2/\log n)$ complexity each.*

Proof. Starting from n initial partitions, the algorithm progressively merges these partitions, halving their number each iteration, requiring a total of $\log n$ iterations. At the iteration i, there are $\frac{n}{2^i}$ partitions, each potentially containing up to 2^i devices. A merged partition P' would have set of devices of size 2^{i+1}, and initial clustering of size $O(1)$. By Proposition 5, the overall complexity of applying Flat FAACL to a new partition is $O(2^i)^2$. As there are total of $\frac{n}{2^i}$ partitions, the per-iteration complexity is $O(2^i)^2 \times \frac{n}{2^i}$=$O(2^i \cdot n)$. Then the overall complexity until convergence is $\sum_{i=1}^{\log n} O(2^i \cdot n)$=$O(n^2)$. □

In summary, the complexity analysis underscores Hierarchical FAACL's efficiency at managing computational resources and adapting to the scale of federated learning environments. In contrast to Flat FAACL, which faces a potential complexity of up to $O(n^3)$ due to its $O(n)$ iterations, Hierarchical FAACL reduces this complexity to $O(n^2)$, ensuring completion within just $\log n$ iterations. By reducing the complexity and leveraging a logarithmic number of iterations, Hierarchical FAACL offers a scalable and efficient solution to clustering in large, distributed networks.

Appendix B

Experiment Environment

B.1 Code

The code for FAACL is available for review in an anonymized github repository: https://github.com/FAACI

B.2 Experimental Setup (Software, Hardware, Randomization)

The implementation was coded in Python. Randomization was done by using three seeds in Numpy. The seeds were set to 10, 55 and 2077 for all the algorithms and datasets. Experiments were run on a single Nvidia GPU (either T4 or A40).

B.3 Model Architecture

The neural architecture used for dataset MNIST, EMNIST, FASHION, and FEMNIST is the Multi-layer Perceptron, a feedforward neural network with two hidden layers for FEMNIST and one hidden layer for the other datasets. We also performed L2 regularization and utilized a ReLU activation function and a softmax output layer with a Sparse Categorical Cross-Entropy loss, that is trained using Stochastic Gradient Descent. For the dataset of Sent140, we use the sequential model, starting with an input layer that expects sequences

of length 25 with 300 features each. The model uses two bidirectional LSTM (Long Short-Term Memory) layers, which are a type of recurrent neural network (RNN) layer suited for learning from sequences. The first LSTM layer has 64 units and returns sequences, feeding into another bidirectional LSTM layer with 32 units that does not return sequences. This is followed by a dense layer with 64 neurons and ReLU activation, a dropout layer with a rate of 0.5 to prevent overfitting, and finally, a dense output layer with 2 neurons and softmax activation for binary classification.

B.4 Hyperparameters

Table B.1: Hyperparameter Summary Table for Scenario S_0, S_1, S_2, A_1, A_2

Parameter	Dataset	S_0	S_1	S_2	A_1	A_2
Epochs	MNIST	300	300	300	300	300
	EMNIST	300	300	300	300	300
	FASHION	300	300	300	300	300
	Cifar10	300	300	300	300	300
Learning rate	MNIST	0.01	0.01	0.01	0.01	0.01
	EMNIST	0.003	0.003	0.003	0.003	0.003
	FASHION	0.005	0.005	0.005	0.005	0.005
	Cifar10	0.002	0.002	0.002	0.002	0.002
δ	MNIST	3	3	3	3	3
	EMNIST	2	2	3	3	3
	FASHION	2	3	3	3	3
	Cifar10	2	3	3	3	3
ϵ	MNIST	0.7	0.7	0.7	0.5	0.4
	EMNIST	0.7	0.7	0.7	0.4	0.1
	FASHION	0.7	0.7	0.7	0.5	0.1
	Cifar10	0.6	0.6	0.6	0.4	0.2

In our methodology for identifying potential merges between clusters C_1 and C_2, we employ a statistical approach where the significance threshold α is compared against the p-value from a statistical test. This test evaluates the null hypothesis $\ell(C_1.\theta, z)+\epsilon < \ell(C_2.\theta, z)$, aiming to determine the likelihood of a merge based on the model parameters θ and data point z.

For the implementation of FedDrift, we introduce a distance metric D_{ij} representing the proximity between cluster i and cluster j. A merge is considered when D_{ij} falls below a predefined threshold δ, indicating a significant overlap in the data representation of both clusters.

Table B.2: Hyperparameter Summary Table for Scenario A_0

Parameter	FEMNIST	Sent140
Epochs	300	300
Learning rate	0.005	0.005
δ	4	4
ϵ	0.5	0.7

The number of Epochs, learning rate, δ, and ϵ are summarized in Table B.1 and B.2 for different experimental scenarios.

Our experiments incorporate both Gaussian and Salt & Pepper noise to construct the experiments with devices having different data quality. Gaussian noise, characterized by its variance and mean, introduces a continuous perturbation, while salt & pepper noise, specified by a density parameter, simulates random pixel corruptions. The configurations for these noise parameters are outlined in Table B.3.

Table B.3: Noise Parameters Summary Table

Parameter	MNIST	EMNIST	FASHION	CIFAR10
Gaussian Noise variance	0.4	1.0	0.9	0.6
Gaussian Noise mean	0.0	0.0	0.0	0.0
Salt &Pepper Noise density	0.7	0.6	0.7	0.4

In the continual learning extension, we specify hyperparameters for different scenarios and datasets as outlined in Table B.4 below. This table summarizes the settings for all scenarios S_1, S_2, A_1, A_2.

Table B.4: Hyperparameter Summary Table for Scenario S_1, S_2, A_1, A_2

Parameter	Dataset	S_1	S_2	A_1	A_2
Epochs	MNIST	100	100	100	100
	FASHION	100	100	100	100
	CIFAR10	100	100	100	100
Learning rate	MNIST	0.01	0.01	0.01	0.01
	FASHION	0.005	0.005	0.005	0.005
	CIFAR10	0.002	0.002	0.002	0.002
Time steps	MNIST	10	10	10	10
	FASHION	10	10	10	10
	CIFAR10	10	10	10	10
δ	MNIST	3	3	3	3
	FASHION	3	3	3	3
	CIFAR10	3	3	3	3
ϵ	MNIST	0.5	0.5	0.3	0.2
	FASHION	0.5	0.5	0.3	0.1
	CIFAR10	0.4	0.4	0.3	0.1