Auxo: Efficient Federated Learning via Scalable Client Clustering

Jiachen Liu
University of Michigan
amberljc@umich.edu

Fan Lai
University of Illinois Urbana-Champaign
fanlai@illinois.edu

Yinwei Dai
Princeton University
yinweid@cs.princeton.edu

Aditya Akella
University of Texas at Austin
akella@cs.utexas.edu

Harsha V. Madhyastha
University of Southern California
madhyast@usc.edu

Mosharaf Chowdhury
University of Michigan
mosharaf@umich.edu

ABSTRACT

Federated learning (FL) is an emerging machine learning (ML) paradigm that enables heterogeneous edge devices to collaboratively train ML models without revealing their raw data to a logically centralized server. However, beyond the heterogeneous device capacity, FL participants often exhibit differences in their data distributions, which are not independent and identically distributed (Non-IID). Many existing works present point solutions to address issues like slow convergence, low final accuracy, and bias in FL, all stemming from client heterogeneity.

In this paper, we explore an additional layer of complexity to mitigate such heterogeneity by grouping clients with statistically similar data distributions (*cohorts*). We propose Auxo to gradually identify such cohorts in large-scale, low-availability, and resource-constrained FL populations. Auxo then adaptively determines how to train cohort-specific models in order to achieve better model performance and ensure resource efficiency. Our extensive evaluations show that, by identifying cohorts with smaller heterogeneity and performing efficient cohort-based training, Auxo boosts various existing FL solutions in terms of final accuracy (2.1%–8.2%), convergence time (up to 2.2×), and model bias (4.8% - 53.8%).

CCS CONCEPTS

• **Computing methodologies → Distributed artificial intelligence**.

KEYWORDS

Federated Learning, Unsupervised Learning

ACM Reference Format:
Jiachen Liu, Fan Lai, Yinwei Dai, Aditya Akella, Harsha V. Madhyastha, and Mosharaf Chowdhury. 2023. Auxo: Efficient Federated Learning via Scalable Client Clustering. In *ACM Symposium on Cloud Computing (SoCC '23), October 30–November 1, 2023, Santa Cruz, CA, USA.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3620678.3624651

1 INTRODUCTION

Federated learning (FL) enables distributed clients to collaboratively train an ML model without centralizing their local data to the cloud. It circumvents the systematic privacy risk and cost of data transfers in centrally collecting user data. Hence, FL is increasingly being adopted by many popular applications, such as Google's Gboard [21], Apple's Siri [57], NVIDIA's medical platform [43], Meta's Ads recommendation [53], and WeBank risk prediction [50].

Federated Learning (FL) typically involves a substantial number of clients, ranging from hundreds to millions, and the training process can span days or even weeks [84]. Given the limited availability and resource constraints of client devices, only a fraction of clients contribute to each round of training in practice. Therefore, it is essential to reduce the training time while accommodating these practical constraints. However, FL encounters unique challenges stemming from statistical heterogeneity among user data, which contributes significantly to extended training time and suboptimal model performance [44, 47, 73, 89]. Several studies that try to mitigate the effect of statistical heterogeneity, such as FedYoGi [59], q-FedAvg [42], FTFA [13], have shown

SoCC '23, October 30–November 1, 2023, Santa Cruz, CA, USA

ACM ISBN 979-8-4007-0387-4/23/11…$15.00
https://doi.org/10.1145/3620678.3624651

that their convergence speed depends on the degree of heterogeneity, both theoretically and empirically (§2.2).

We explore the possibility of mitigating this issue at its core by grouping clients with similar data distributions, known as *cohorts* [86] (§2.3). If a population has K cohorts, training K separate models – one for each cohort with lower statistical heterogeneity – can boost the performance of many existing FL algorithms that are complementary to ours and focus on convergence [38, 41, 59], fairness optimization [42], communication efficiency [3, 31, 65], etc.

Although recent works attempted to identify cohorts and train separate models for them [11, 22, 47, 83], they are not applicable to real-world FL deployments. This is because unlike easy-to-deploy solutions such as FedAvg and FedYoGi [51, 59], clustering clients at scale and in the wild poses unique challenges (§2.4). Existing solutions often ignore the scale and sparsity of the device participation. They also ignore the constraints on availability and capacity of end-user devices, which calls for low-overhead algorithms.

We propose **Auxo** to enable 1) scalable cohort identification to reduce intra-cohort heterogeneity in large-scale and limited-availability FL scenarios; and 2) efficient cohort-based training to facilitate most FL optimizations, such as faster training completion and better model accuracy, without additional resource requirements. Auxo addresses the following challenges toward practical FL deployment (§4). First, unlike existing clustering strategies which require exhaustive passes through all clients [67], on-demand device availability [11], or additional on-device training for every participant [16, 22], Auxo introduces a more flexible client clustering solution. It allows sporadic client availability, respects client resource constraints, and maintains client privacy. Auxo can progressively identify cohorts and scalably cluster clients based on their gradients in spite of the absence of anchored gradients for straightforward comparison. Second, unlike expensive and ad-hoc hyper-parameter tuning stages used in existing solutions, Auxo progressively generates the appropriate number of cohorts and identifies suitable timings to create them. Thus, Auxo maximizes the use of limited client resources to enhance training speed and model performance.[1] Finally, we design a scalable system to support efficient cohort clustering and training at scale while being robust to uncertainties (e.g., failure tolerance and unfavorable settings) at scale (§5).

We have implemented (§6) and evaluated (§7) Auxo on a wide variety of real-world FL datasets, tasks, and algorithms at scale. Compared to existing solutions, Auxo improves the performance for various FL algorithms, such as better model accuracy (2.1%-8.2%) and convergence speed (up to 2.2×) and smaller bias of model accuracy (4.8% - 53.8%).

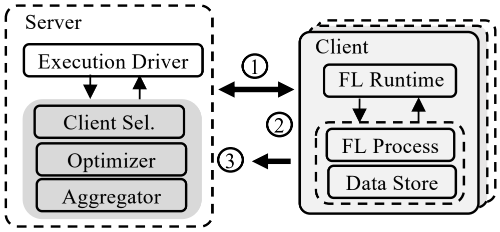


Figure 1: Traditional FL overview. The server first selects from available clients and sends out model weights. Clients train the updated model on their local dataset. After training is finished, clients report their model gradient to the server.

Overall, we make the following contributions in this paper:

(1) We propose a systematic clustering mechanism to identify cohorts for the practical large-scale, low-availability and resource-constrained FL setting.
(2) We identify a sweet spot for jointly optimizing model convergence and training cost, and provide analytical insights to ensure good model performance.
(3) We implement and evaluate Auxo at scale, showing large improvements in final accuracy, convergence time, and model fairness over the state-of-the-art. Auxo is open-source and available on GitHub.[2]

2 BACKGROUND AND MOTIVATION

We start with a brief introduction of federated learning (§2.1), followed by the challenges it faces in real-world settings (§2.2) . Next, we describe some opportunities to improve FL that motivates our work (§2.3). Finally, we explain the limitations of related works that motivate our algorithm and system design (§2.4).

2.1 Federated Learning

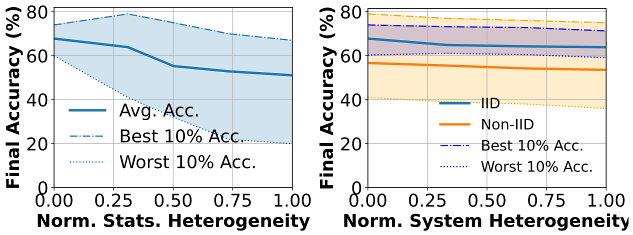
A typical cross-device FL system consists of two primary components (Figure 1): A logically centralized cloud *server* that maintains a single global model and many distributed *clients* with private local data. The overall lifecycle of an FL training round can be divided into three broad stages.

① **Selection stage:** Clients check in with the server continuously to announce their availability for FL computation. The server selects a number of *participants* for that round based on its client selection strategy.

② **Execution stage:** The selected participants download the current model from the server and perform server-specified computation on their local data.

[1] We refer to the number of participants that contribute to a round of FL training as training resource throughout this paper.

[2] https://github.com/SymbioticLab/FedScale/tree/master/examples/auxo



(a) Statistical heterogeneity.

(b) System heterogeneity.

Figure 2: The impact of heterogeneity on final accuracy.

③ **Aggregation stage:** Participants that successfully complete the execution stage send model updates back to the server. The server aggregates the updates to finalize an updated model for the next round.

2.2 Heterogeneity Challenges in FL

Unlike centralized ML, FL faces unique challenges in terms of statistical and system heterogeneity. The former refers to the varying data volumes and difference of data distribution across clients, which hinders model convergence; the latter refers to variations in system characteristics among participants' devices, which results in large differences in training performance. Increasing heterogeneity in either dimension leads to poor performance.

Impact of statistical heterogeneity. Under large statistical heterogeneity across clients, poor model accuracy, training time and fairness are often exacerbated, because the model is deployed on individual clients but is often trained over all the clients. Existing works that address statistical heterogeneity in FL assume bounded heterogeneity to simplify the problem complexity [41, 44, 59, 89]. However, we notice this does not hold in practical FL settings, which leads to great performance degradation under larger statistical heterogeneity.[3] Indeed, our analysis of FedYoGi [59] (a state-of-the-art FL algorithm) on OpenImage [56] (an FL image dataset), in Figure 2a shows that the model accuracy and its fairness across clients worsens with increasing statistical heterogeneity. To achieve the same model performance under larger heterogeneity, more communication and/or computation costs are needed. This is true for personalization algorithms as well [73].

Impact of system heterogeneity. Heterogeneity of system-level characteristics raise challenges such as fault tolerance and straggler mitigation [30, 41]. Over-commitment [10], which discards updates from slowest-responding participants, is commonly used to reduce the impact of stragglers, but it may lead to participation bias against slow devices. Figure 2b shows the final accuracy of the OpenImage task under different degrees of system heterogeneity (variance of system speed). For each experiment, we control the round duration and the number of successful participants to be the same; as a result, participation bias exacerbates with increasing system heterogeneity. Since participation bias may enhance statistical heterogeneity in another form, the final accuracy decreases with increasing system heterogeneity (albeit at a slower rate than statistical heterogeneity).

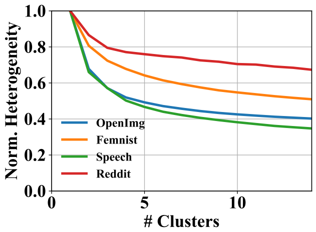


Figure 3: Intra-cluster heterogeneity in real datasets.

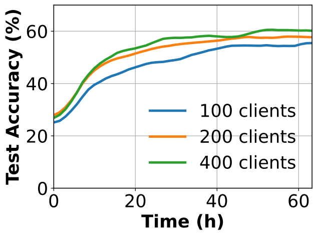


Figure 4: Diminishing return when adding more participants.

2.3 Opportunities

The opportunity for improving FL training performance, therefore, lies in decreasing heterogeneity especially the statistical heterogeneity based on the observation of the previous subsection. By identifying statistically homogeneous groups and performing FL within each group, we may be able to boost model performance of most FL algorithms that are suffered by the heterogeneity.

Despite large statistical heterogeneity across the entire FL client population, there exist groups of statistically similar clients in most large populations. Figure 3 shows that for four representative FL workloads [36] in the real world. We use K-means clustering (with increasing values of K) on clients' data distribution by their L2-distance metric. As the number of clusters increases from one (i.e., traditional FL with one global model) to larger values, we observe a small number of statistically similar groups emerge for most datasets.

However, training K models to converge may need more training resources compared to training one model. As shown in Figure 4, increasing training resources has diminishing returns on the model convergence, which presents the primary opportunity leveraged in this work: *instead of letting all available clients contribute to a single global model, it may be more beneficial to partition them into several cohorts, each with smaller heterogeneity.*

2.4 Limitations of Existing Clustered FL

Recent efforts in the ML community have (theoretically) explored to create smaller groups of statistically similar clients.

[3] In this experiment, we measure the statistical heterogeneity among a set of clients using the popular L2 distance on their data distributions [38].

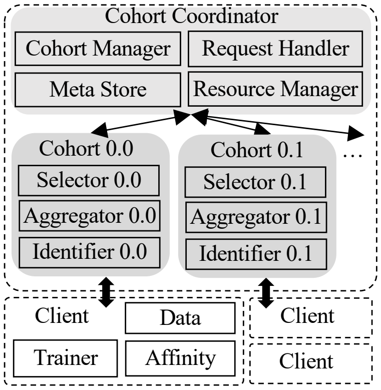


Figure 5: Auxo architecture. Auxo server guides Auxo clients to train on their best-fit cohorts.

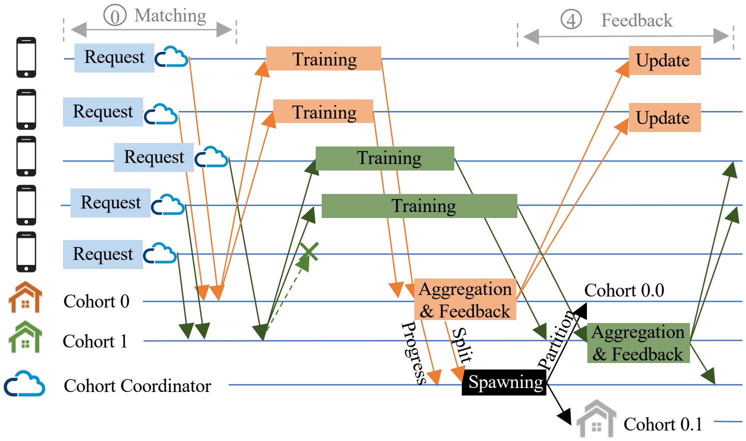


Figure 6: Auxo lifecycle. Clients check in with their affinity requests, and participate training within matched cohorts. Cohorts are gradually identified based on participants response.

	CFL	FL+HC	FlexCFL	IFCA	Auxo
Partial part.	×	✓	✓	✓	✓
Low avail.	×	×	✓	✓	✓
Res. constraint	×	×	×	×	✓
Training perf.	×	×	×	×	✓

Table 1: Comparing Auxo with existing Clustered FL.

Yet, existing clustered FL algorithms often fall short across multiple dimensions in practical deployments, which motivates us to design systems support for efficient cohort identification and training. We empirically show the superior performance of Auxo over them too (§7.2).

Scalability. FL in practice often involves millions of clients, and only a small fraction (~5% [10, 36]) are available to participate in during a time window. Such low availability and partial participation limit the available information for clustering algorithms. This, unfortunately, is ignored by CFL [67], multi-center [47] and FL+HC [11], making their deployment impractical as they require a complete pass over the entire population to identify clusters. Furthermore, clients usually have limited on-board resources, but IFCA [22], FlexCFL [16], ICFL [83], k-FED [15] and FL+HC require extra computation for *every* client to assign them to a cluster. This imposes a significant computational and communication burden on already resource-constrained devices and diverts resources away from the primary task of model training. For example, IFCA initiates multiple global models and broadcasts all models for each participant to choose from in each round; and FlexCFL and FL+HC require pre-training for every client to identify their clusters.

Efficiency. In addition to the challenge of identifying statistically similar groups at scale, how to leverage those similar groups to improve model performance introduces new trade-offs in deciding the right number of cohorts and time to partition. Given a fixed amount of resources, generating more cohorts results in smaller heterogeneity; but it divides up the fixed training resource and unique training data per cohort, which hurts model convergence and generalizability. Moreover, partitioning clients too early can lead to model bias as the model is not generalized well by training on various clients, while partitioning too late can result in model variance over high heterogeneity. Unfortunately, most existing clustered FL algorithms are unaware of these tradeoffs, and rely on ad-hoc hyper-parameter tuning, which is prohibitively expensive as FL training can take many days and consume a large amount of resources.

In conclusion, as detailed in Table 1, an effective client clustering solution in Federated Learning (FL) should take into account the following realistic constraints:

(1) Partial participation: The algorithm should accommodate FL training that involves only a fraction of total participants in each round.
(2) Low availability: The algorithm should respect clients' sporadic availability, without necessitating participation from any clients at a specified time.
(3) Resource constraints: The algorithm should avoid demanding additional on-device computation for performing clustering.
(4) Training performance: The algorithm should optimize model performance—focusing on convergence and generalizability—within the constraints of a fixed training

resource. This includes consideration of how clustering the FL population might positively impact performance despite reduced heterogeneity.

3 AUXO OVERVIEW

Auxo progressively reduces the intra-group heterogeneity and improves the model performance through cohort identification and cohort-based training toward practical FL. In this section, we introduce the cohort abstraction, provide an overview of how Auxo manages cohorts in a distributed fashion and fits into the FL life cycle.

3.1 Cohort Abstraction

Instead of training only one global model, Auxo trains a model separately for each group of clients that shares similar statistical data characteristics. We refer to each of these groups, which can perform independent FL training over more homogeneous clients than the overall population, as a *cohort* $C_m (m \in [1, M])$ with two associated properties:

(1) A cohort should hold a specialized model that targets on it data distribution with smaller heterogeneity.
(2) A cohort should have enough members $|C_m|$ to form a meaningful group and deliver the benefit of partition.

Traditional (i.e., cohort-agnostic) FL training has a single cohort with unbounded heterogeneity among the members.

3.2 Auxo Architecture

Auxo server consists of two primary components (Figure 5):

(1) A logically centralized **cohort coordinator** performs three main functions. First, it manages existing cohorts for fault tolerance. Second, it matches clients to their best-fit cohorts. Finally, it monitors the progress of cohort training and identification in order to decide cohort partition when it observes an opportunity for better model convergence .
(2) A set of **cohorts** each performs independent FL training. Each cohort contains traditional FL components such as aggregator and client selector. On top of traditional FL training activities, each cohort continuously identifies its internal composition, reports its progress to the coordinator and waits for the partition instruction from the coordinator.

FL Lifecycle in Auxo. As shown in Figure 6, following the traditional FL stages in Section 2.1, Auxo adds a matching stage ⓪ and a feedback stage ④ before and after the traditional round.

⓪ **Matching stage:** When checking in, clients using Auxo optionally include an affinity request (a hint about their cohort preference) to the cohort coordinator. If it took part in the training of one or more cohorts in the past, its preference is dependent on previous feedback. Otherwise, it has no preference. The cohort coordinator forwards the affinity request to the corresponding cohort based on its search algorithm and client's request.

① - ③ **Traditional FL stages:** Each cohort starts a traditional FL training round independently after continuously receiving its client requests from the cohort coordinator. These traditional stages include client selection, client training, server aggregation, and so on.

④ **Feedback stage:** After the traditional FL round finishes, each cohort updates the affinity feedback for its current participants based on the Auxo clustering algorithm (§4). Then, each participant receives an affinity feedback – w.r.t. the cohort it trained with — and updates the corresponding affinity record for submitting requests in a future round of FL training. During this stage, each cohort also reports its training and identification progress to the cohort coordinator.

Resource management: Auxo jointly maximizes model convergence and resource efficiency in two ways. First, its scalable cohort identification algorithm does not require extra on-device computation and uses the same amount of resources as traditional FL algorithms(§4.1- §4.3). Second, it carefully chooses the number of cohorts and time to partition to theoretically guarantee better model convergence and generalizability despite each cohort having less training resources than the previous global model (§4.4).

Threat model and robustness. Like state-of-the-art production FL systems [10, 30, 75], Auxo considers an *honest-but-curious* centralized server for aggregation, which can infer any information without interfering with the FL training. Auxo also assumes that most clients are honest (correct), and only a small fraction can act maliciously under the control of a bad actor [70]. We elaborate on how Auxo can provide robustness under this threat model in Section 5.2.

4 AUXO CLUSTERING

In this section, we present the core clustering algorithm used in Auxo to identify cohorts (§4.1- §4.3). Then, we introduce the systems techniques to enable cohort-based training under realistic constraints (§4.4).

4.1 Problem Formulation and Overview

Auxo aims to accurately cluster clients by their statistical heterogeneity into appropriate cohorts under the following real-world FL constraints:

(1) *Scalability:* The participants $\mathcal{P}^r$ in each round are only a small fraction of all clients (N), i.e., $|\mathcal{P}^r| \ll N$. How to identify cohorts and cluster clients at scale under such low client availability?

(2) *Resource Efficiency:* How to conduct the clustering process without incurring overhead on devices, such as extra model training and client participation that do not contribute to model training?

(3) *Information Deficiency:* The information available to today's FL central server is limited to such as gradients and training loss. How to cluster clients without requesting additional information from clients?

Problem Formulation: The *input* to the server is a list of participants along with their gradients collected over training rounds based on these two constraints. Intuitively, the gradient of client relies on its local dataset x_i and the received model weights (unique for the round r and cohort m), and this gradient is multi-dimensional, embedding more information than its counterparts (e.g., training loss). As such, we can formulate the *input* of the clustering algorithm in each round r as $\{\{g^r_m(x_i)\}_{i \in \mathcal{P}^r_m}\}_{m \in [1, M_r]}$, where $g^r_m(x_i)$ is the gradient of participant i , $\mathcal{P}^r_m$ is the participants list, and M_r is the number of cohorts.

The *output* is the cohort membership $\{S_i \in [1, M]\}$ for each client $i \in [1, N]$. Following the objective of traditional clustering algorithms [48], Auxo also aims to minimize the average intra-cohort heterogeneity (J) defined as:

$$J = \sum_{m=1}^{M} \frac{1}{2|\{x|S_x = m\}|} \sum_{S_i, S_j = m} ||x_i - x_j||^2. \qquad (1)$$

Intuitively, we can model it as a clustering problem $\{x_1, ..., x_N\} \rightarrow \{S_1, ..., S_N\}$, whereas doing so encounters new challenges.

(1) How to derive client data similarity without direct access to data and without iterating all but part of the clustering objects every round.

(2) How to assign new incoming clients to the best-fit cohort without prior information after Auxo generates more than one cohorts.

Following this problem definition and challenge, Algorithm 1 illustrates the overview of Auxo clustering mechanism, which consists of an online cluster algorithm to cluster clients at scale (§4.2) and the cohort selection for individual FL clients (§4.3). Note that, Auxo's clustering algorithm can operate in the background, imposing no additional overhead on the training process.

Algorithm 1: Auxo Clustering Algorithm

```
Input: Participants list $\mathcal{P}$, Exploration factor $\epsilon$
Output: Client-cohort membership list $S_{\mathcal{D}}$
$M \leftarrow 1$;                         ▷ Initialize the number of cohorts.
$S_{\mathcal{D}} \leftarrow 0$;                ▷ Initialize client-cohort membership.
$R_{\mathcal{D},M} \leftarrow 0$;              ▷ Initialize client-cohort reward.
$L_{\mathcal{D},M} \leftarrow N/A$.            ▷ Initialize client-cohort cluster id.
for each round $r = 1, 2, ...$ do
    $\mathcal{P}^r_m = \{i | S_i = m, i \in \mathcal{P}^r\}$
    for each cohort $m = 1, ..., M$ in parallel do
        $R_{\mathcal{P}^r_m}$ = Auxo-Clustering ($\mathcal{P}^r_m$ )
    $S_{\mathcal{P}^r_m}$ = CohortSelection($R_{\mathcal{P}^r_m}, \epsilon, r$)
return $S_{\mathcal{D}}$

Function ClientClustering(Participants list $\mathcal{P}^r_m$):
    /* Identify clusters on the fly. (4.4)                    */
    if $r == 1$ then
        $L_{\mathcal{P}^r_m,m}$ = Kmeans($g^r_m(x_{\mathcal{P}^r_m}), K$).
    else
        $\mathcal{P}_k = \{i | L_{i,m} = k, i \in \mathcal{P}^r_m\}, \forall k \in [0, K)$
        $C_k = \overline{\{g^r_m(x_{\mathcal{P}_j})}, \forall k \in [0, K)$
        $L_{\mathcal{P}^r_m,m} = arg\min_k ||g^r_m(x_{\mathcal{P}^r_m}) - C_k||_2$
    /* Decide partitioning to start separate training. (4.2) */
    if PartitionCriteria(m) then
        $M = M + K - 1$
        $R_{\mathcal{D},m+k} = R_{\mathcal{D},m} + 0.1 * \mathbb{1}(L_{\mathcal{D},m} == k), \forall k \in$
          $[0, K)$
    /* Update rewards for cohort selection. (4.4)             */
    if $M > 1$ then
        $R_{\mathcal{P}^r_m,m}$ = ExploitReward($R_{\mathcal{P}^r_m,m}, x_{\mathcal{P}^r_m}$)
        $R_{\mathcal{P}^r_m,m'}$=ExploreReward($R_{\mathcal{P}^r_m}, m'$), $\forall m \neq m'$
    return $R_{\mathcal{P}^r_m}$

Function CohortSelection( (Reward list $R_{\mathcal{P}^r_m}, \epsilon, r$)):
    for client i in $\mathcal{P}^r_m$ do
        if random(0,1) > $\epsilon^r$ then
            $S_i$=random(0, M)
        else
            $S_i = \arg\max R_i$
    return $S_{\mathcal{P}^r_m}$
```

4.2 Online Clustering

Auxo resorts to the similarity of clients' gradients to capture their statistical similarity. Our design is inspired by the recent advances in ML theory [67, 69], which show that the data heterogeneity can attribute to the gradient divergence [41] and a smaller heterogeneity would have smaller gradient divergence for the *same* initial model weight. Here, we measure such gradient divergence using the widely-used cosine similarity [82] among the input batch of gradients $g^r_m(x_i), i \in \mathcal{P}^r_m$ to investigate client similarity.[4] Compared to other counterparts such as L-2 distance which does not take into account the direction of the gradients, cosine similarity better quantifies how similarly their needed model changes are directed.

[4] Cosine similarity measures the similarity between two vectors of an inner product space [82].

However, the sporadic participation of clients in each training round limits the data available for clustering algorithms to a subset of the entire client population at any given time. Traditional clustering algorithms, such as K-means and KNN, require a complete pass of the population, rendering them inapplicable here. Mini-batch clustering algorithms [68], on the other hand, operate on small batches of the population each round, maintain a running centroid for cluster assignments. Nonetheless, this strategy cannot directly be applied in our case because we only know the gradients $g_m^r(x_{\mathcal{P}m^r})$ and not the raw data $x_{\mathcal{P}}$. Further, since the gradient $g_m^r(\cdot)$ depends on the initial model of round r and client data - both unknown and different across rounds and cohorts. These complexities preclude us from maintaining absolute cluster centroids over successive rounds in a straightforward manner, making naive mini-batch clustering infeasible.

Algorithm 1 outlines how Auxo starts with one cohort for the entire FL population, and then adaptively identifies cohorts based on gradients of mini-batch clients. After using K-means to initialize the cluster prototype (Line 15), in each round, Auxo collects the training feedback from the clients and assigns clients to their closest clusters (Line 17). Meanwhile, Auxo incrementally refines cluster centers based on the gradients of newly assigned clients in each round (Line 18). With repeated cluster updating and clients assignment, Auxo can effectively identify the clusters at scale (Line 19). Each new cohort starts with the parent cohort model weights with the same architecture, performs conventional FL steps separately, and converges to different model weights. Once discernible clusters emerge and certain partition criteria are fulfilled (e.g., enough participants left for model convergence after partition), Auxo decides to spawn cohorts based on these pre-identified clusters (Line 20) and train cohort models separately within their corresponding client groups. At runtime, Auxo adaptively decides the right time and the right number of cohorts to partition to find the sweet spot of model performance and the resource consumption of training multiple cohorts (§ 4.4).

4.3 Cohort Selection

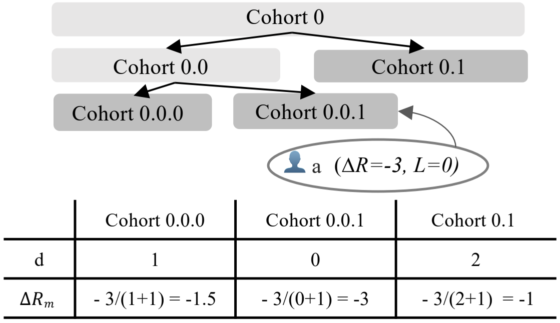
Although clustering captures the membership of already-identified clients, doing so for a new client is unkown a priori, since we neither have access to client data nor have absolute cohort centers that can inform a new client to choose the closest cohort. This challenge is further amplified by the large training population, wherein more FL clients participate in model training for the first time than not.

To address this, Auxo adopts an *exploration-exploitation* strategy to efficiently identify the cohort membership for new participants (Line 11). This allows us to first randomly assign a new client to a cohort. After getting the feedback on how well the client fits in that cohort, Auxo attempts to identify a more suitable cohort for it the next time it participates again.

Auxo uses reward-based decaying ϵ-greedy selection [72] to help the client find the best-fit cohort (Line 11). With an aim to maximize the expected reward for each client, there is a $1 - \epsilon$ probability of selecting a cohort with a maximum reward and a ϵ probability of selecting cohorts randomly, where $\epsilon \in [0, 1]$ is the exploration factor that decays over time to account for the latest information. Intuitively, smaller gradient divergence compared to the members within the explored cohort means a better fit and gives a higher reward. Hence, Auxo calculates the relative divergence between the client gradients and the explored cohort center. This is done by first estimating the cohort center via averaging the client gradients within the cohort $\mathcal{P}^r_{m,Known}$, to be $D = ||g_m^r(x_{\mathcal{P}^r_m}) - \overline{g_m^r(x_{\mathcal{P}^r_{m,Known}})}||_2$, where $\overline{g_m^r(x_{\mathcal{P}^r_{m,Known}})}$ represents the estimated cluster centers for cohort m. Next, we take the popular approach to identify outlier clients [4]. Specifically, we consider clients as outliers if their distance to the cohort center exceeds the threshold, which is calculated as the sum of the mean and the standard deviation of D, denoted as $avg(D) + std(D)$. If the client gradient distance to the cohort center is larger than this threshold, this client is not considered as the cohort member. As such, the instant reward becomes $\Delta R = 1 - \frac{1}{avg(D)+std(D)} D$, where the client with a negative ΔR would be considered as an outlier of the cohort. Then, Auxo updates the reward between each client and its explored cohort with a decay factor γ as $R_{\mathcal{P}^r_m,m} = \gamma * \Delta R + (1 - \gamma) * R_{\mathcal{P}^r_m,m}$, γ=0.2 by default in popular exploration-exploitation designs.

Efficient cohort exploration. During exploration, there may exist multiple cohorts for a client to try out with. To improve the searching efficiency and save device training resources, during both training and deployment, Auxo enables a new client to perform a binary search to find the most appropriate cohort by predicting the rewards for other unexplored cohorts m' through function `ExploreReward()` (Line 25): $R_{\mathcal{P}^r_m,m'} \mathrel{+}= \frac{R_{\mathcal{P}^r_m}}{d(m,m')+1}, \forall m \neq m'$.

The intuition behind the cohort search is that the client may perform similar to or receive similar rewards from the cohorts that are closer/similar to the previously explored ones, and vice versa. To find out the cohort similarity, we first define the distance (d) between two cohorts to be the distance to their lowest common ancestral cohorts based on the hierarchical cluster relationship among cohorts. Given an explored cohort m and the reward ΔR_m for a participant, Auxo calculates the distance d and updates the rewards for unexplored cohorts to be inversely proportional to their distance. For example, if a client receives a negative reward for the



	Cohort 0.0.0	Cohort 0.0.1	Cohort 0.1
d	1	0	2
ΔR_m	- 3/(1+1) = -1.5	- 3/(0+1) = -3	- 3/(2+1) = -1

Figure 7: Reward update based on the hierarchical structure among all cohorts.

chosen cohort, then he is more likely to explore another furthest cohort with higher reward given by `ExploreReward()` next time.

Taking Figure 7 as an example, a new client a explores $Cohort_{0.0.1}$ and receives the corresponding feedback rewards -3. Then, with the intuition that the client may have similar performance on a closer cohort, Auxo calculates the distance between $Cohort_{0.0.1}$ and other cohorts. As shown in Table 7, Auxo updates the rewards to be inversely proportional to the cohort distance $d_{m'}$: $\Delta R_{m'} = \frac{\Delta R}{d_{m'}+1}$ Since $Cohort_{0.0.1}$ and $Cohort_{0.1}$ have a larger distance between them, $Cohort_{0.1}$ ends up with a relatively higher reward and has higher probability to be explored by client a in the future.

4.4 Cohort-Based Training

While clustering reduces heterogeneity within a cohort, generating a larger number of cohorts may dilute available resources for each individual cohort when operating under fixed training resources. Consequently, this leads to a new trade-off between resource efficiency and model convergence. As shown in Figure 3, generating more cohorts has diminishing returns in terms of heterogeneity. In a setting where total training resources are fixed, allocating resources to a larger number of cohorts implies fewer resources for each, which may negatively affect model convergence. Conversely, having too few cohorts is insufficient for adequately addressing intra-cohort heterogeneity. Therefore, Auxo faces the challenge of optimally determining both the number of cohorts and the timing for their creation to balance resource efficiency and model performance effectively.

Intuitively, the decision to generate new cohorts should be based on the extent of client heterogeneity and the available training resource budget post-partition. When client heterogeneity is significant and the resource budget is sufficient, the creation of additional cohorts is warranted to further reduce client heterogeneity. On the other hand, when these conditions are not met, the creation of new cohorts should be deferred.

We next provide analytical insights to ground our strategy. Prior works in ML theory [32, 44, 59, 89] have shown that the convergence rate of FL training is largely dominated by heterogeneity. We start by analyzing the relationship between heterogeneity and training resources in theory. Inspired by the convergence analysis of FedAvg [32], we establish the following Lemma. More detailed proof are available in Appendix A.

Lemma 4.1. *If the population and training resources are partitioned into up to K cohorts, to theoretically achieve better model convergence, intra-cohort heterogeneity should be reduced by $\sqrt{K}$ times when the training resource $|\mathcal{P}|$ is larger than $\alpha\sqrt{\frac{|\mathcal{P}_0|}{J_0^2}}$. α is a constant setting that elaborates the relationship between model convergence and training resources.*

From Lemma 4.1, we notice that the number of generated cohorts rely on the expected reduced heterogeneity and a lower bound of training resources. As such, Auxo actively monitors the gradient divergence within each cohort at runtime to estimate the potential heterogeneity reduction.

When a sufficient decrease (e.g., $\frac{1}{\sqrt{K}}$) in intra-cohort heterogeneity and ample post-partition training resources are detected, Auxo autonomously partitions the population into a maximum of K cohorts, allotting equal training resources to each. This strategy theoretically enhances model convergence through cohort-based training in Auxo. As for some FL datasets with larger heterogeneity, FL developers can further improve model convergence by dynamically raising the resource budget to allow generating more cohorts.

In addition to deciding the right number of cohorts, the time to cohort partition is also critical to model convergence. As cohort partitioning may reduce the unique training data for each cohort model, the trade-off between model bias and variance can be affected by the time of partition. On the one hand, hard partitioning of the entire population at the beginning could reduce heterogeneity, but it could also reduce the amount of unique training data for each cohort model, leading to poor model generalizability. On the other hand, late partitioning exposes the model to diverse training data but leads to worse model variance due to high heterogeneity. These also guide the *reuse* of identified cohorts to facilitate other FL tasks.

From the sensitive analysis of cohort partition time (§7.4), we found the model convergence is not sensitive to exact partition time as long as cohorts are not partitioned at the beginning or the end of the training. We report more results about the effect of partition time on model convergence in Section 7.4.

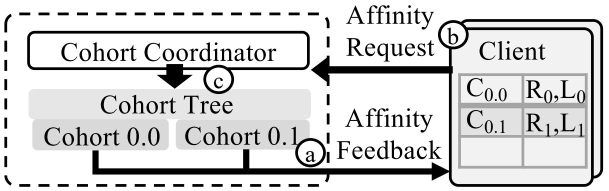


Figure 8: Scalable Design Overview a. Cohort affinity feedback. b. Client affinity request. c. Cohort coordinator request match.

Finally, the start time of gradient-based clustering can impact the efficiency of the process. In the early stages of training, gradients are often large and may not adequately capture the distributional features of the data. However, as the model approaches convergence, the gradients become more informative indicators of data similarities. Thus, it is crucial for Auxo to judiciously select the optimal starting point for clustering so as not to delay the cohort identification. Detailed results discussing the effect of the clustering start time on model convergence can be found in Section 7.4.

5 AUXO SYSTEM DESIGN

In this section, we discuss how to design a practical and robust system on top of the clustering algorithm under real-world challenges.

5.1 Distributed Auxo

As the scale of training grows, the server faces more server challenges for tremendous storage, fault tolerance, and client privacy in order to maintain the cohort and client information. Thus, Auxo designs a solution to use a soft-state server that offloads cohort-related information to individual clients to mitigate these challenges. In this subsection, we describe how to implement the proposed clustering algorithm in a distributed fashion, while achieving the same objective.

Firstly, we introduce *affinity message*, which is a lightweight message containing all necessary state information needed to identify cohorts in a distributed fashion. Affinity message consists of two pieces of information between a client and a cohort to enable efficient state transmission: (*Reward* $R \in \mathbb{R}$, *Cluster index* $L \in [0, K)$). The *reward* implies how well the client fits for this cohort. The *cluster index* expresses the client's cluster membership within this cohort and is used to indicate its future cohort index.

Through exchanging affinity messages between different components, Auxo encourages similar clients to collaborate more in a distributed fashion. As shown in Figure 8, we next describe ⓐ how a cohort informs its relationship with its participants, ⓑ how clients request for their preferred cohort based on the affinity feedback and ⓒ how the cohort coordinator matches different requests.

Affinity Feedback. At the end of each round, every cohort computes the affinity feedback to inform participants about their relationship with the cohort. These affinity feedback correspond to the clustering results returned by Algorithm 1 Line 8 (reward R) and Line 19 (cluster index L). These clustering results would be sent back to the participants respectively in the format of affinity messages, which informs the participant about whether the cohort is a good fit and which sub-cohort to select after partitioning.

Client Reaction . After receiving the affinity feedback from the cohort, the client would update its affinity records itself based on the equation in Algorithm 1 Line 24- 25 and copy the cluster index directly. Following the same decaying $\epsilon-$greedy selection method (§4.3), clients select the cohort to train by themselves. Then, clients ready to participate would submit the corresponding affinity request to the cohort coordinator.

Request Match. After receiving the affinity request, the cohort coordinator matches each client to the cohort it requests. Note that only the leaf cohort in the cohort tree would be returned as it conducts actual FL training inside. The requested cohort may not be the leaf cohort because some clients may not be aware of the cohort partitioning, which is not yet transparent to all clients. In this case, the cohort coordinator should assist clients to select their best-fit cohort through finding the closest leaf cohort indicated by the requested cohort and cluster index in the affinity message.

After finding a proper cohort, cohort coordinator would forward this affinity request to the corresponding cohort to initiate traditional FL rounds. Moreover, these forwarded affinity requests provide each cohort with all necessary input to conduct the clustering algorithms. Thus, after receiving the gradients from its participants, each cohort is able to run the Algorithm 1 independently to compute the aforementioned affinity feedback.

5.2 Resilient Auxo

Fault Tolerance. Auxo enables fast recovery to minimize the impact on the model training. Upon a cohort process failure in the server, the cohort coordinator spawns a new cohort process. The new cohort loads the model from the latest checkpoint and restarts the incomplete round.

If the cohort coordinator fails, cohort processes would continue their current independent FL training and wait until a new cohort coordinator to be re-spawned. Clients checking in within that recovery period would be ignored.

Finally, Auxo is resilient to client failures just like traditional FL by design. Most client failure handlers, which are orthogonal to Auxo, can be applied directly. In addition, a failed client, who may lose its own affinity records, would restart exploring again. We empirically show that Auxo can tolerate a certain amount of such client failures while continuing to benefit the FL training (§7.5).

Robustness. Based on Auxo's threat model, Auxo can naturally cooperate with some existing privacy-preserving approaches [19, 52, 54] to address potential threats from both the server and clients. To handle the honest-but-curious server, Auxo can be used with local differential privacy (LDP), which is used to provide user-level privacy guarantees. Since differential privacy is immune to post-processing [18] and Auxo's clustering is post-processing, Auxo incurs no additional privacy loss.

To handle a small fraction of unreliable clients [6, 8], Auxo can be used with existing adversary-resilient solutions [12, 71]. For Auxo-specialized adversaries, such as fake affinity requests, Auxo detects anomalies by comparing its position in the cluster with its claimed affinity (Algorithm 1 Line 8). If a significant discrepancy is detected, Auxo will detect and blacklist it. In Section 7.5, we empirically evaluate Auxo's robustness under these scenarios.

6 IMPLEMENTATION

We have implemented Auxo as an independent Python library (1, 664 lines) to serve existing FL frameworks (e.g., TFF [1] and PySyft [2]), and integrated it with FedScale [36] for evaluations. Auxo abstracts away the cohort identification and partition so that FL developers can easily try out their FL algorithms or datasets on top of Auxo without any modifications.

Auxo's implementation consists of the three components described in Section 3: The cohort coordinator manages and spawns cohort processes, which initiate FL training tasks. Clients continuously submit their training requests based on their availability and affinity records. Then, the cohort coordinator takes client training requests as input and forwards the requests to corresponding cohorts. Each cohort process conducts conventional FL training with the assigned available clients independently. At the end of each individual round, the Auxo clustering algorithm runs within every cohort and reports clustering results to each participant over the network. All training metadata and model weights are checkpointed periodically for fault tolerance. Meanwhile, the cohort coordinator continuously monitors the progress of cohorts for resource management and failure recovery.

7 EVALUATION

We evaluate Auxo's effectiveness for six different ML tasks as well as different choices of FL algorithms. Our evaluation shows the following key highlights:

Dataset	#Clients	#Samples
Google Speech [76]	2,618	105K
FEMNIST [14]	3,400	640K
OpenImage-Easy	10,133	1M
OpenImage [56]	13,771	1.3M
Amazon Review [33]	42,031	2M
Reddit [60]	63,058	5M

Table 2: Statistics of the six datasets in evaluation.

(1) Auxo speeds up model convergence on different FL datasets up to 2.2×, while improving final test accuracy by 3.4%-8.2%. Auxo cooperates with existing FL efforts (e.g., personalization) and boosts final test accuracy by 2.1%–6.7%. Auxo can mitigate model bias across devices by 4.8% and 53.8% and improve resource efficiency (§7.2).
(2) Auxo outperforms existing clustered FL solutions up to 4.8× in time and 5.2× in resources (§7.3).
(3) Auxo performs well across a broad range of its parameter settings (§7.4).

7.1 Experiment Setup

Evaluation environment. We use 24 NVIDIA Tesla P100 GPUs on CloudLab [17] to emulate the large-scale client training in our evaluations. The client data distribution follows the real-world partition, where client data can vary in quantities, data, and label distribution. We use the open-source benchmark FedScale [36] with standardized setup including realistic device capacity, data, and client availability traces. We report the simulated wall clock time by relying on these realistic FL system and data traces.

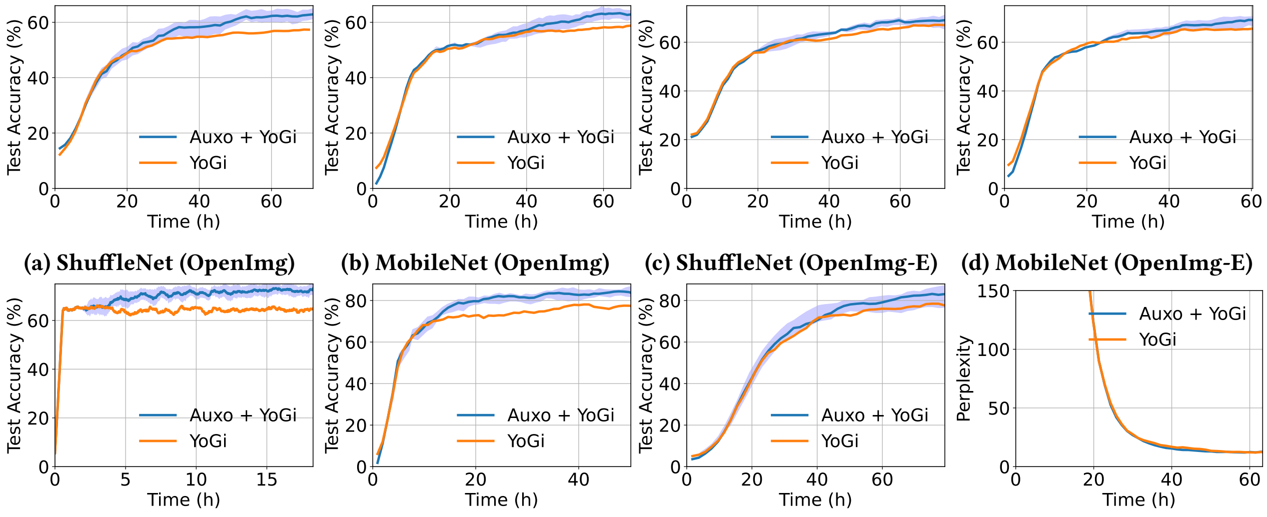
Datasets and models. We run three categories of applications with six FL datasets [36] of different scale factors using real-world partitions, whose statistics are reported in Table 2. The clients for all datasets can check-in with Auxo multiple times following the availability trace.

(1) *Speech Recognition*: We train Resnet-34 [27] on a small-scale Google Speech dataset with 35 commands.
(2) *Image Classification*: We train Resnet-18 on small-scale FEMNIST with 62 handwritten digits to classify. Also, we train ShuffleNet [87] and MobileNet [66] on middle-scale OpenImage with 596 classes to classify, whereas OpenImage-Easy only has 60 classes.
(3) *Language Modeling*: We train logistic regression (LR) on middle-scale Amazon Review for ratings prediction, and Albert model [39] on large Reddit for word prediction.

These applications are widely used in real end-device applications [80], and these models are lightweight.

Task	Dataset	Model	Target Acc.	Auxo Speedup	Auxo Acc. Impr.
Image Classification	FEMNIST	ResNet-18	82.2%	1.2×	7.3%
	OpenImg	MobileNet	56.5%	1.3×	4.8%
		ShuffleNet	58.2%	2.2×	5.0%
	OpenImg-Easy	MobileNet	65.4%	1.4×	3.4%
		ShuffleNet	64.8%	1.2×	4.4%
Language Modeling	Amazon Review	Logistic Regression	65.3%	1.2×	8.2%
	Reddit	Albert	7 perplexity	1×	0 perplexity
Speech Recognition	Google Speech	ResNet-34	78.5%	1.5×	5.7%

Table 3: Summary of improvements on time to accuracy. We target the highest accuracy attainable by YoGi.



(a) ShuffleNet (OpenImg) (b) MobileNet (OpenImg) (c) ShuffleNet (OpenImg-E) (d) MobileNet (OpenImg-E)

(e) LR (Amazon) (f) ResNet (FEMNIST) (g) ResNet (Google Speech) (h) Albert (Reddit)

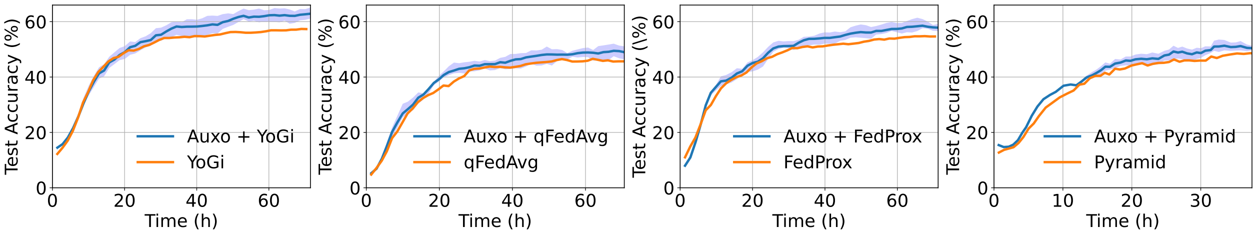
Figure 9: Time-to-Accuracy performance on different dataset. For the language modeling (LM) task, a lower perplexity is better. The solid line reflects the average test accuracy. The shaded portion covers the test accuracy performance among all cohorts generated by Auxo.

Parameters. We follow the standardized experiment and parameter settings in FedScale. We adopt an over-commitment strategy to mitigate stragglers which allow 25% failures or stragglers every round as in real FL deployments [10]. We set the number of participants per round to be 200, the local minibatch size to be 6, and the initial learning rate to be 4e-5 for the Albert model, and 0.05 for other models. And we use the linear scaling rule [23] to scale the learning rate.

Metrics. The *time-to-accuracy* performance, *final test accuracy*, and *model bias* are our key metrics. We use the cohort member's test data, which follows the realistic data partition, to evaluate each cohort model. The test data would be the global test data if we end up with one global model. For each experiment, we report the average top-1 accuracy based on the results over 3 runs.

7.2 End-to-End Performance

Auxo's performance on different datasets. We first evaluate Auxo's performance on different real-world FL datasets. In the following experiments, we adopt YoGi as the FL algorithm because it outperforms other FL algorithms most of the time. Table 3 summarizes the key time-to-accuracy performance of all datasets, where we tease apart the overall improvement with statistical and system ones. We quantify the time-to-accuracy as speedup by Auxo, which measures how many times Auxo can speed up to achieve the target accuracy compared to the baseline time cost. Figure 9 reports the



(a) Yogi **(b) q-Fedavg** **(c) FedProx** **(d) PyramidFL + YoGi**

Figure 10: Auxo with different FL algorithms.

timeline of training to achieve different accuracy, as different cohorts perform FL asynchronously. The shaded portion covers the test accuracy among all cohorts generated by Auxo.

We notice that Auxo speeds up the wall-clock time to reach target accuracy up to 2.2× faster. Moreover, the final accuracy for different datasets is improved by 3.4%–8.2%. The benefit of Auxo varies over datasets. For most of the datasets, Auxo can achieve significant final accuracy improvement. Nevertheless, Auxo does not improve Reddit task because the clients' texting behavior is similar to each other that makes it hard to identify significant groups as shown in Figure 3. Hence, Auxo decides not to partition into multiple cohorts to maximize the benefit of clustering in FL training.

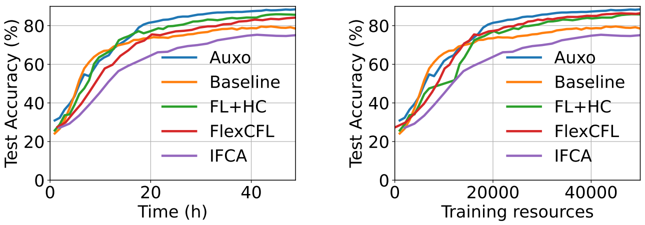
Auxo's performance on different FL algorithms. We then evaluate Auxo's performance on ShuffleNet-OpenImage with different FL Algorithms, which are complementary to Auxo. We refer to YoGi running atop Auxo as YoGi+Auxo, and similarly for FedProx, q-FedAvg and PyramidFL+YoGi.

As shown in Figure 10, Auxo speeds up the time to reach the target accuracy of baseline algorithms, from 1.2× to 2.2× faster and improve the final test accuracy by 3%–6.8%.

As for the personalization algorithm FTFA, we adopt the cohort models generated by Auxo to conduct local training using FTFA on corresponding cohort members. In addition to faster convergence of the initial model, Auxo also improves the average test accuracy of FTFA from 63.18% to 67.40% with local fine-tuning.

Auxo's benefit on resource efficiency. We finally show that Auxo can optimize resource efficiency on OpenImg dataset by saving 55% training resources. We also account for the affinity maintaining overhead into the client resource usage, which is around 0.02% of the total resource consumption.

Auxo's benefit on model bias. We show that Auxo can also mitigate the model bias due to smaller intra-cohort statistical heterogeneity. We report the variance of the final accuracy distributions, the worst and best 10% test accuracy in Table 4. Our experiment show that the variance of test accuracy is decreased for all the datasets by 4.8% and 53.8%.



(a) Accuracy-Time **(b) Accuracy-Resource**

Figure 11: Comparison with clustered FL (FEMNIST).

7.3 Clustered FL Comparison

We compare Auxo with four existing clustered FL algorithms CFL, FL+HC, FlexCFL, and IFCA in terms of three metrics: time-to-accuracy, resource-to-accuracy, and final accuracy. Since these algorithms do not meet some real-world FL constraints (Table 1), we simplify the settings accordingly.

We compare with CFL in small-scale settings (∼ 100 clients) from the FEMNIST dataset to meet their full participation assumption. We observe little difference between CFL, Auxo, and baseline (i.e., no cohorts) in terms of time and resources used due to the absence of significant clusters within small populations. However, this highlights the need for large-scale FL settings, where CFL cannot even be applied as it does not support partial participation.

To compare with FL+HC, FlexCFL and IFCA, we conduct experiments with the full FEMNIST and Amazon Review datasets *without* the client availability traces to align with their constraints. As shown in Table 5 and Figure 11, Auxo achieve better time efficiency 1.4×−4.8× and better resource efficiency 1.3 × −4.8× compared to the related works especially for the large-scale Amazon dataset, due to our efficient and scalable algorithm design. Also, our result for IFCA are consistent with Motley [77].

7.4 Sensitivity Analysis

Impact of different degrees of heterogeneity. We generate different statistical heterogeneity by applying affine shift [61]

Dataset	Setting	Worst 10% (%)	Best 10% (%)	Variance
OpenImg	Auxo	38	83	267
	Baseline	34	79	296
OpenImg -Easy	Auxo	38	88	234
	Baseline	33	84	273
Review	Auxo	50	100	460
	Baseline	32	100	995
FEMNIST	Auxo	63	97	171
	Baseline	60	93	185
Speech	Auxo	57	100	479
	Baseline	52	100	503

Table 4: Summary of improvements on model bias.

		FL+HC	FlexCFL	IFCA	Auxo
FEMNIST	Speedup	1.7×	1.3×	0.5×	2.4×
	Efficiency	1.6×	1.8×	0.5×	2.4×
	Final acc.	5.8%	7.1%	1.3%	9.1%
Amazon Review	Speedup	0.4×	0.4×	0.5×	2.3×
	Efficiency	0.4×	0.5×	0.5×	2.1×
	Final acc.	-2.9%	-2.7%	0.6%	5.4%

Table 5: Summary of improvements over baseline (i.e., no cohorts) in terms of time, resource and accuracy.

on OpenImage, then we evaluate Auxo with YoGi across different degrees of statistical heterogeneity. Figure 13a reports the final test accuracy as well as top 10% and worst 10% client test accuracy on different degrees of heterogeneity. We observe that Auxo can improve model accuracy and mitigate model bias under different degrees of heterogeneity. Moreover, similar to the previous experiment, Auxo achieves faster time-to-accuracy performance from 1.2× to 1.8×.

Impact of time to partition. We investigate the impact of different cohort partition times on the model convergence. As mentioned in Section 4.4, the partition time relates to the trade-off between model generalizability and intra-cohort heterogeneity. As shown in Figure 13b, we choose different partition times with the same cohort composition and report the test accuracy to time performance on FEMNIST. We observe that cohort-based training all outperform the baseline experiment with one cohort. However, early partitions such as FlexCFL and IFCA are worse than intermediate partitions, because it sacrifices the model generalizability. Similarly, late partition after convergence as CFL does not outperform intermediate partition, because it slows down the model convergence to a smaller heterogeneous population.

Impact of time to start clustering. We investigate the impact of the clustering start time on the model convergence. As mentioned in Section 4.4, different clustering start time may affect the clustering accuracy and efficiency. To quantify how well the gradient similarity correlates with data similarity, we use Pearson correlation coefficient $r = \frac{\sum_{i=1}^{n}(G_i-\bar{G})(D_i-\bar{D})}{\sqrt{\sum_{i=1}^{n}(G_i-\bar{G})^2\sum_{i=1}^{n}(D_i-\bar{D})^2}}$, where D and G are pairwise data similarity and gradient similarity. As shown in Figure 12, the similarity correlation slightly increases over training rounds, which suggests a slightly later cluster start time.

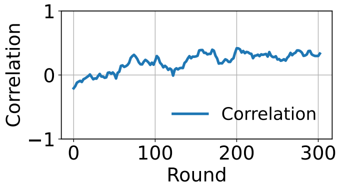


Figure 12: Impact of cluster start time.

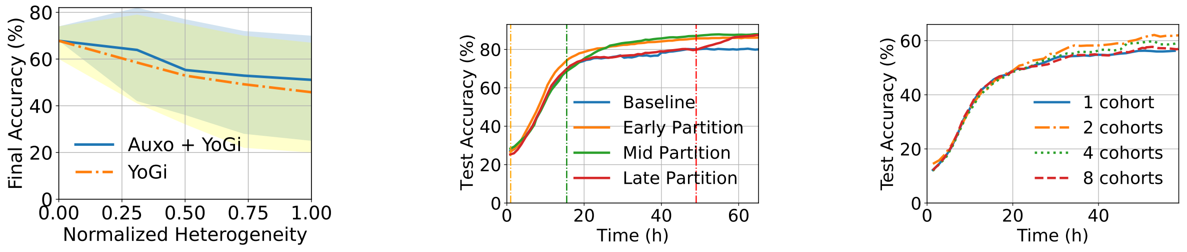
Impact of the number of cohorts. We investigate the impact of the number of cohorts generated by Auxo, which relates to the trade-off between training resources and intra-cohort heterogeneity (§4.4). As shown in Figure 13c, we observe that the model convergence is negatively affected once the number of cohorts exceeds 4 under the same resource budget. By further comparing the reduce of heterogeneity with different number of cohorts indicated in Figure 3, this result verifies that better model convergence can be achieved as long as the heterogeneity can proportionally compensate the reduced training resources

7.5 Auxo Resilience

Impact of differential privacy. Auxo is robust to local differential privacy (LDP). LDP is used to protect user-level privacy by adding Gaussian noise to the client update before sending it to the server, but it hurts model accuracy. We evaluate Auxo's performance under LDP for a learning task on the Amazon Review dataset. To achieve (ϵ, δ)-differential privacy, where $\delta = 10^{-6}$ based on the training scale and $\epsilon = 2, 4, 8$, we set the noise scale $\sigma = 1.0, 0.77, 0.6$. As shown in Figure 14a, Auxo can still benefit FL training across different differential privacy guarantees.

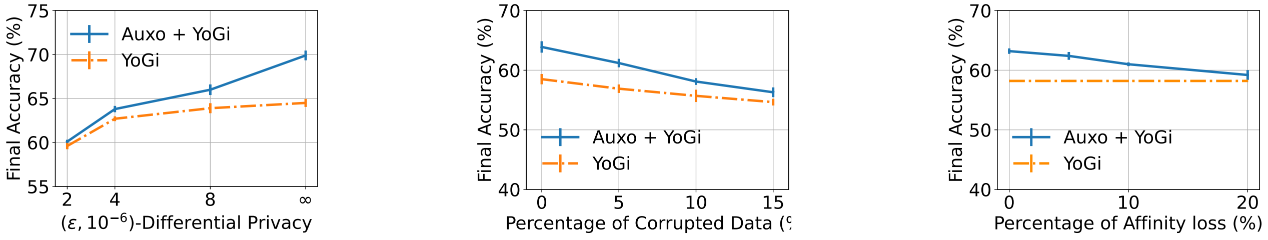
Impact of malicious attacks. We investigate the robustness of Auxo by manually involving corrupted clients. Following a popular adversarial ML setting that introduces local model poisoning [20], we randomly flip the ground-truth data labels for these corrupted clients. As shown in Figure 14b, we introduce different percentages of corrupted clients to the OpenImg task. We set the percentage of corruption below 15%, which is a practical percentage under the real-world setting [20]. We observe that Auxo still improves performance across different degrees of corruption through identifying malicious clients and eliminating their interference.

Impact of unstable client. Finally, we show Auxo is robust with unstable clients who fail to maintain their affinity records, which may result in less accurate clustering results.



(a) Different degrees of heterogeneity. **(b) Different partition time.** **(c) Different number of cohorts.**

Figure 13: Sensitivity analysis.



(a) Local DP. **(b) Corrupted client.** **(c) Client affinity loss.**

Figure 14: Robustness of Auxo under different scenarios.

We consider loss rates from 0% to 20% and report the corresponding final test accuracy in Figure 14c. We notice Auxo outperforms the baseline across different affinity loss rates.

8 RELATED WORK

Distributed Machine Learning. Distributed ML in data centers has been well-studied [49, 55, 85], where homogeneous data and workers are assumed [24]. With the same training goal, FL raises its unique challenges including the data heterogeneity and system heterogeneity. As a result, Auxo aims at speeding up the training process through directly reducing the intra-cohort heterogeneity at scale.

Federated Learning. FL is a distributed machine learning paradigm [10, 34] with two key challenges: statistical and system heterogeneity. State-of-the-art FL algorithms try to tackle these two challenges and optimize different targets including model convergence [38, 41, 45, 46, 59, 88], fairness [40, 42], privacy [9, 62–64], efficiency [5, 26, 51, 79], and robustness [25, 40]. However, they underperform in FL because they do not tackle the root cause of FL challenges but mitigate the negative effect caused by heterogeneity.

Federated Analytics. There has been significant work on geo-distributed data analytics [29, 35, 58, 74]. They mainly optimize the execution latency [37] and resource efficiency [28, 78]. To further preserve privacy for distributed data, Orchard [64] and Honeycrisp [63] have been proposed to enable large-scale differentially private analytics. Helen [91] and Cerebro [90] allow multiple parties to securely train models without revealing their data.

Traditional Clustering Algorithms. Clustering algorithms [7, 81] are used in popular data mining techniques, which usually assume access to all data. However, under FL setting, it is non-trivial to design a clustering algorithm because of the unavailability of data. Auxo proposes a clustering algorithm that can be applicable to the FL settings.

FL Client Clustering. In order to leverage the nature of clusters in real-world FL dataset, many algorithms have been proposed to identify the clusters among FL clients. However, existing clustered FL solutions [11, 16, 22, 67] mainly suffer from scalability and practicality, which are hard to adapt to large-scale, low-participation, and resource-constraint FL training. Considering all real-world constraints, Auxo build a practical system to identify cohorts and benefit FL training.

9 CONCLUSION

We presented Auxo, which builds on top of the observation that there exist natural groups of statistically similar clients (cohorts) in large real-world FL populations. Auxo identifies cohorts with reduced intra-cohort heterogeneity at scale, addressing heterogeneity-borne FL challenges at their roots. Auxo proposes an efficient algorithm and practical system that can be applied under real-world FL constraints to significantly benefit FL training in terms of model convergence, final accuracy, and model bias.

ACKNOWLEDGMENTS

We are grateful to the anonymous reviewers, our shepherd Matthias Boehm and SymbioticLab members for their valuable comments and suggestions that improved the paper. We thank the CloudLab team for providing GPU servers for Auxo experiments. This work was supported in part by NSF grant CNS-2106184 and a grant from Cisco.

REFERENCES

[1] 2018. TensorFlow Federated. https://www.tensorflow.org/federated.

[2] 2019. PySyft. https://github.com/OpenMined/PySyft.

[3] Ahmed M. Abdelmoniem, Atal Narayan Sahu, Marco Canini, and Suhaib A. Fahmy. 2023. Resource-Efficient Federated Learning. (2023).

[4] Vaibhav Aggarwal, Vaibhav Gupta, Prayag Singh, Kiran Sharma, and Neetu Sharma. 2019. Detection of Spatial Outlier by Using Improved Z-Score Test. In *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*. 788–790. https://doi.org/10.1109/ICOEI.2019.8862582

[5] Dan Alistarh, Jerry Li, Ryota Tomioka, and Milan Vojnovic. 2016. QSGD: Randomized Quantization for Communication-Optimal Stochastic Gradient Descent. In *NeurIPS*.

[6] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2020. How To Backdoor Federated Learning. In *AISTATS*.

[7] P. Berkhin. 2006. *A Survey of Clustering Data Mining Techniques*. Springer Berlin Heidelberg.

[8] Battista Biggio, Blaine Nelson, and Pavel Laskov. 2012. Poisoning Attacks against Support Vector Machines. In *ICML*.

[9] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *CCS*.

[10] Kallista A. Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé Kiddon, Jakub Konečný, Stefano Mazzocchi, H. Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. 2019. Towards Federated Learning at Scale: System Design. In *MLSys*.

[11] Christopher Briggs, Zhong Fan, and Péter András. 2020. Federated learning with hierarchical clustering of local updates to improve training on non-IID data. *IJCNN* (2020).

[12] Yudong Chen, Lili Su, and Jiaming Xu. 2019. Distributed Statistical Machine Learning in Adversarial Settings: Byzantine Gradient Descent. *ACM SIGMETRICS* (2019).

[13] Gary Cheng, Karan N. Chadha, and John C. Duchi. 2021. Fine-tuning is Fine in Federated Learning. *CoRR* abs/2108.07313 (2021). arXiv:2108.07313 https://arxiv.org/abs/2108.07313

[14] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. 2017. EMNIST: Extending MNIST to handwritten letters. In *IJCNN*.

[15] Don Kurian Dennis, Tian Li, and Virginia Smith. 2021. Heterogeneity for the Win: One-Shot Federated Clustering. In *ICML*.

[16] Moming Duan, Duo Liu, Xinyuan Ji, Yu Wu, Liang Liang, Xianzhang Chen, and Yujuan Tan. 2021. Flexible Clustered Federated Learning for Client-Level Data Distribution Shift. *IEEE TPDS* (2021).

[17] Dmitry Duplyakin, Robert Ricci, Aleksander Maricq, Gary Wong, Jonathon Duerig, Eric Eide, Leigh Stoller, Mike Hibler, David Johnson, Kirk Webb, Aditya Akella, Kuangching Wang, Glenn Ricart, Larry Landweber, Chip Elliott, Michael Zink, Emmanuel Cecchet, Snigdhaswin Kar, and Prabodh Mishra. 2019. The Design and Operation of CloudLab. In *ATC*.

[18] C. Dwork and A. Roth. 2014. *The Algorithmic Foundations of Differential Privacy*. https://books.google.com/books?id=Z3p8swEACAAJ

[19] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *ACM SIGSAC*.

[20] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. 2020. Local Model Poisoning Attacks to Byzantine-Robust Federated Learning. *SEC* (2020).

[21] GBoard. 2020. Federated Learning: Collaborative Machine Learning without Centralized Training Data. https://ai.googleblog.com/2017/04/federated-learning-collaborative.html. Accessed August 29, 2021.

[22] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. 2020. An Efficient Framework for Clustered Federated Learning. In *NeurIPS*.

[23] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. 2017. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677* (2017).

[24] Juncheng Gu, Mosharaf Chowdhury, Kang G. Shin, Yibo Zhu, Myeongjae Jeon, Junjie Qian, Hongqiang Liu, and Chuanxiong Guo. 2019. Tiresias: A GPU Cluster Manager for Distributed Deep Learning. In *NSDI*.

[25] Hanxi Guo, Hao Wang, Tao Song, Yang Hua, Zhangcheng Lv, Xiulang Jin, Zhengui Xue, Ruhui Ma, and Haibing Guan. 2021. Siren: Byzantine-robust federated learning via proactive alarming. In *Proceedings of the ACM SoCC*.

[26] Peizhen Guo, Bo Hu, and Wenjun Hu. 2021. Mistify: Automating DNN Model Porting for On-Device Inference at the Edge. In *NSDI*.

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *CVPR*.

[28] Charlie Hou, Kiran K Thekumparampil, Giulia Fanti, and Sewoong Oh. 2018. Reducing the communication cost of federated learning through multistage optimization. (2018).

[29] Kevin Hsieh, Aaron Harlap, Nandita Vijaykumar, Dimitris Konomis, Gregory R. Ganger, Phillip B. Gibbons, and Onur Mutlu. 2017. Gaia: Geo-Distributed Machine Learning Approaching LAN Speeds. In *NSDI*.

[30] Dzmitry Huba, John Nguyen, Kshitiz Malik, Ruiyu Zhu, Michael G. Rabbat, Ashkan Yousefpour, Carole-Jean Wu, Hongyuan Zhan, Pavel Ustinov, Harish Srinivas, Kaikai Wang, Anthony Shoumikhin, Jesik Min, and Mani Malek. 2022. Papaya: Practical, Private, and Scalable Federated Learning. In *MLSys*.

[31] Zhifeng Jiang, Wei Wang, Baochun Li, and Bo Li. 2022. Pisces: Efficient Federated Learning via Guided Asynchronous Training. In *SoCC*.

[32] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. 2020. SCAFFOLD: Stochastic Controlled Averaging for Federated Learning. In *ICML*.

[33] Phillip Keung, Yichao Lu, György Szarvas, and Noah A. Smith. 2020. The Multilingual Amazon Reviews Corpus. *CoRR* abs/2010.02573 (2020). arXiv:2010.02573 https://arxiv.org/abs/2010.02573

[34] Nicolas Kourtellis, Kleomenis Katevas, and Diego Perino. 2020. Flaas: Federated learning as a service. In *Proceedings of the 1st workshop on distributed machine learning*.

[35] Fan Lai, Mosharaf Chowdhury, and Harsha Madhyastha. 2018. To Relay or Not to Relay for Inter-Cloud Transfers?. In *HotCloud*. USENIX Association.

[36] Fan Lai, Yinwei Dai, Sanjay S. Singapuram, Jiachen Liu, Xiangfeng Zhu, Harsha V. Madhyastha, and Mosharaf Chowdhury. 2022. FedScale: Benchmarking Model and System Performance of Federated Learning at Scale. In *ICML*.

[37] Fan Lai, Jie You, Xiangfeng Zhu, Harsha V. Madhyastha, and Mosharaf Chowdhury. 2020. Sol: Fast Distributed Computation Over Slow Networks. In *NSDI*.

[38] Fan Lai, Xiangfeng Zhu, Harsha V. Madhyastha, and Mosharaf Chowdhury. 2021. Oort: Efficient Federated Learning via Guided Participant Selection. In *OSDI*.

[39] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *ICLR*.

[40] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. 2021. Ditto: Fair and Robust Federated Learning Through Personalization. In *ICML*.

[41] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. In *MLSys*.

[42] Tian Li, Maziar Sanjabi, Ahmad Beirami, and Virginia Smith. 2020. Fair Resource Allocation in Federated Learning. In *ICLR*.

[43] Wenqi Li, Fausto Milletarì, Daguang Xu, Nicola Rieke, Jonny Hancox, Wentao Zhu, Maximilian Baust, Yan Cheng, Sébastien Ourselin, M. Jorge Cardoso, and Andrew Feng. [n.d.]. Privacy-Preserving Federated Brain Tumour Segmentation. Springer-Verlag.

[44] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. 2020. On the Convergence of FedAvg on Non-IID Data. In *ICLR*.

[45] Zitao Li, Bolin Ding, Ce Zhang, Ninghui Li, and Jingren Zhou. 2021. Federated matrix factorization with privacy guarantee. *VLDB* (2021).

[46] Junxu Liu, Jian Lou, Li Xiong, Jinfei Liu, and Xiaofeng Meng. [n.d.]. Projected federated averaging with heterogeneous differential privacy. *VLDB* ([n. d.]).

[47] Guodong Long, Ming Xie, Tao Shen, Tianyi Zhou, Xianzhi Wang, and Jing Jiang. 2022. Multi-center federated learning: clients clustering for better personalization. *WWW* (2022).

[48] J MacQueen. 1967. Classification and analysis of multivariate observations. In *5th Berkeley Symp. Math. Statist. Probability*. University of California Los Angeles LA USA, 281–297.

[49] Luo Mai, Guo Li, Marcel Wagenländer, Konstantinos Fertakis, Andrei-Octavian Brabete, and Peter Pietzuch. 2020. KungFu: Making Training in Distributed Machine Learning Adaptive. In *OSDI*.

[50] Brendan McMahan and Daniel Ramage. 2017. Utilization of FATE in Risk Management of Credit in Small and Micro Enterprises. https://www.fedai.org/cases/utilization-of-fate-in-risk-management-of/credit-in-small-and-micro-enterprises/. Accessed August 31, 2021.

[51] H. B. McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *AISTATS*.

[52] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2018. Learning Differentially Private Recurrent Language Models. In *ICLR*.

[53] Meta. 2021. What Are Privacy-Enhancing Technologies (PETs) and How Will They Apply to Ads? https://about.fb.com/news/2021/08/privacy-enhancing-technologies-and-ads/. Accessed Nov 25, 2021.

[54] Lorenzo Minto, Moritz Haller, Benjamin Livshits, and Hamed Haddadi. 2021. *Stronger Privacy for Federated Collaborative Filtering With Implicit Feedback*.

[55] Deepak Narayanan, Aaron Harlap, Amar Phanishayee, Vivek Seshadri, Nikhil R. Devanur, Gregory R. Ganger, Phillip B. Gibbons, and Matei Zaharia. 2019. PipeDream: Generalized Pipeline Parallelism for DNN Training. In *SOSP*.

[56] OpenImg. 2018. Google Open Images Dataset. https://storage.googleapis.com/openimages/web/index.html.

[57] Matthias Paulik, Matt Seigel, Henry Mason, Dominic Telaar, Joris Kluivers, Rogier C. van Dalen, Chi Wai Lau, Luke Carlson, Filip Granqvist, Chris Vandevelde, Sudeep Agarwal, Julien Freudiger, Andrew Byde, Abhishek Bhowmick, Gaurav Kapoor, Si Beaumont, Áine Cahill, Dominic Hughes, Omid Javidbakht, Fei Dong, Rehan Rishi, and Stanley Hung. 2021. Federated Evaluation and Tuning for On-Device Personalization: System Design & Applications. *CoRR* abs/2102.08503 (2021).

[58] Qifan Pu, Ganesh Ananthanarayanan, Peter Bodik, Srikanth Kandula, Aditya Akella, Paramvir Bahl, and Ion Stoica. 2015. Low Latency Geo-Distributed Data Analytics. In *SIGCOMM*.

[59] Sashank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and Hugh Brendan McMahan. 2021. Adaptive Federated Optimization. In *ICLR*.

[60] Reddit. 2021. Reddit Comment Data. https://files.pushshift.io/reddit/comments/.

[61] Amirhossein Reisizadeh, Farzan Farnia, Ramtin Pedarsani, and Ali Jadbabaie. 2020. Robust Federated Learning: The Case of Affine Distribution Shifts. In *NeurIPS*.

[62] Edo Roth, Karan Newatia, Yiping Ma, Ke Zhong, Sebastian Angel, and Andreas Haeberlen. 2021. Mycelium: Large-Scale Distributed Graph Queries with Differential Privacy. In *SOSP*.

[63] Edo Roth, Daniel Noble, Brett Hemenway Falk, and Andreas Haeberlen. 2019. Honeycrisp: Large-scale Differentially Private Aggregation Without a Trusted Core. In *SOSP*.

[64] Edo Roth, Hengchu Zhang, Andreas Haeberlen, and Benjamin C. Pierce. 2020. Orchard: Differentially Private Analytics at Scale. In *OSDI*.

[65] Daniel Rothchild, Ashwinee Panda, Enayat Ullah, Nikita Ivkin, Ion Stoica, Vladimir Braverman, Joseph Gonzalez, and Raman Arora. 2020. FetchSGD: communication-efficient federated learning with sketching. ICML.

[66] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *CVPR*.

[67] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. 2021. Clustered Federated Learning: Model-Agnostic Distributed Multitask Optimization Under Privacy Constraints. *IEEE Transactions on Neural Networks and Learning Systems* (2021).

[68] D. Sculley. 2010. Web-Scale k-Means Clustering. In *WWW*.

[69] Ozan Sener and Silvio Savarese. 2018. Active Learning for Convolutional Neural Networks: A Core-Set Approach. In *ICLR*.

[70] Virat Shejwalkar, Amir Houmansadr, Peter Kairouz, and Daniel Ramage. 2022. Back to the Drawing Board: A Critical Evaluation of Poisoning Attacks on Federated Learning. *IEEE SP* (2022).

[71] Jinhyun So, Başak Güler, and A Salman Avestimehr. 2020. Byzantine-resilient secure federated learning. *IEEE Journal on Selected Areas in Communications* (2020).

[72] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[73] Xueyang Tang, Song Guo, and Jingcai Guo. 2022. Personalized Federated Learning with Clustered Generalization. *IJCAI* (2022).

[74] Raajay Viswanathan, Ganesh Ananthanarayanan, and Aditya Akella. 2016. CLARINET: WAN-Aware Optimization for Analytics Queries. In *OSDI*.

[75] Ewen Wang, Ajay Kannan, Yuefeng Liang, Boyi Chen, and Mosharaf Chowdhury. 2023. FLINT: A Platform for Federated Learning Integration. In *MLSys*.

[76] Pete Warden. 2018. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. In *arxiv.org/abs/1804.03209*.

[77] Shanshan Wu, Tian Li, Zachary Charles, Yu Xiao, Ziyu Liu, Zheng Xu, and Virginia Smith. 2022. Motley: Benchmarking Heterogeneity and Personalization in Federated Learning. *FL-NeurIPS* (2022).

[78] Zhe Wu, Michael Butkiewicz, Dorian Perkins, Ethan Katz-Bassett, and Harsha V. Madhyastha. 2013. SPANStore: Cost-Effective Geo-Replicated Storage Spanning Multiple Cloud Services. In *SOSP*.

[79] Yuexiang Xie, Zhen Wang, Dawei Gao, Daoyuan Chen, Liuyi Yao, Weirui Kuang, Yaliang Li, Bolin Ding, and Jingren Zhou. 2023. Federatedscope: A flexible federated learning platform for heterogeneity. *VLDB* (2023).

[80] Mengwei Xu, Jiawei Liu, Yuanqiang Liu, Felix Xiaozhu Lin, Yunxin Liu, and Xuanzhe Liu. 2018. When Mobile Apps Going Deep: An Empirical Study of Mobile Deep Learning. *CoRR* abs/1812.05448 (2018).

[81] Rui Xu and D. Wunsch. 2005. Survey of clustering algorithms. *IEEE Transactions on Neural Networks* (2005).

[82] Ye Xue, Diego Klabjan, and Yuan Luo. 2022. Aggregation Delayed Federated Learning. *IEEE International Conference on Big Data* (2022).

[83] Yihan Yan, Xiaojun Tong, and Shen Wang. 2023. Clustered Federated Learning in Heterogeneous Environment. *IEEE Transactions on Neural Networks and Learning Systems* (2023).

[84] Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. 2018. Applied federated learning: Improving google keyboard query suggestions. *arXiv preprint arXiv:1812.02903* (2018).

[85] Peifeng Yu, Jiachen Liu, and Mosharaf Chowdhury. 2021. Fluid: Resource-Aware Hyperparameter Tuning Engine. In *MLSys*.

[86] Honglin Yuan, Warren Richard Morningstar, Lin Ning, and Karan Singhal. 2022. What Do We Mean by Generalization in Federated Learning?. In *ICLR*.

[87] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. 2018. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. In *CVPR*.

[88] Yuchen Zhao, Payam Barnaghi, and Hamed Haddadi. 2022. Multimodal Federated Learning on IoT Data. In *2022 IEEE/ACM Seventh International Conference on Internet-of-Things Design and Implementation (IoTDI)*.

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

[90] Wenting Zheng, Ryan Deng, Weikeng Chen, Raluca Ada Popa, Aurojit Panda, and Ion Stoica. 2021. Cerebro: A Platform for Multi-Party Cryptographic Collaborative Learning. In *USENIX Security*.

[91] Wenting Zheng, Raluca Ada Popa, Joseph E. Gonzalez, and Ion Stoica. 2019. Helen: Maliciously Secure Coopetitive Learning for Linear Models. In *IEEE S&P*.

A PROOF OF LEMMA 4.1

We first make precise some definitions that are related to the proof from SCAFFOLD and then see the proof of Lemma 4.1.

A.1 Additional definitions

Assumption 1. $g_i(w)$ is unbiased stochastic gradient of f_i with bounded variance, where f_i represents the loss function on client i.

$$\mathbb{E}_{x_i}[||g_i(w) - \nabla f_i(w)||^2] \leq \sigma^2, \forall i, x.$$

where w is the aggregated server model. Note that σ only bounds the variance within clients not across clients.

Assumption 2. $\{f_i\}$ are β-smooth and satisfy:

$$||\nabla f_i(w) - \nabla f_i(v)|| \leq \beta||w - v||, \forall i, w, v.$$

Assumption 3. f_i is μ-convex for $\mu \geq 0$ and satisfies:

$$\langle \nabla f_i(w), v - w\rangle \leq -(\nabla f_i(w) - \nabla f_i(v) + \frac{\mu}{2}||w - v||^2), \forall i, w, v.$$

Assumption 4. (G, B)-BGD or Bounded Gradient Similarity: there exist constants $G \geq 0$ and $B \geq 1$ such that

$$\frac{1}{N}\sum_{i=1}^{N}||\nabla f_i(w)||^2 \leq G^2 + B^2\nabla f(w)), \forall w.$$

A.2 Theoretical Results

Lemma 1. If the population and training resources are partitioned into up to K cohorts, to theoretically achieve better model convergence, intra-cohort heterogeneity should be reduced by $\sqrt{K}$ times when the training resource $|\mathcal{P}|$ is larger than $\alpha\sqrt{\frac{|\mathcal{P}_0|}{J_0^2}}$. α is a constant setting specified in SCAFFOLD that elaborates the relationship between model convergence and training resources.

Proof. We first borrow the proof of convergence analysis on FedAvg (Theorem 1) from SCAFFOLD following the same assumptions mentioned above:

$$\mathbb{E}[f(w^R)] - f(w^*) \leq 3||w^0 - w^*||^2\mu e^{-\frac{\widetilde{\eta}}{2}R}$$
$$+\widetilde{\eta}(\frac{2\sigma^2}{kP}(1 + \frac{P}{\eta_g^2}) + \frac{8G^2}{P}(1 - \frac{P}{N})) + \widetilde{\eta}^2(36\beta G^2),$$
$$\forall\frac{1}{\mu R} \leq \widetilde{\eta} \leq \frac{1}{8(1 + B^2)\beta}$$

where P denotes the training resources, k is the number local steps, η_l is the local step-size, η_g is the global step-size and $\widetilde{\eta} = k\eta_l\eta_g$ is the effective step-size

Since we only care about the effect of training resources P and heterogeneity G on the convergence analysis, we further simplify the right hand side equation to be

$$h(P, G) = \frac{\theta}{P} + \gamma\frac{G^2}{P} + \rho G^2 + \xi$$

where θ, γ, ρ and ξ are constant settings. Since we proportionally partition the population and training resources, we can assume $(1 - \frac{S}{N})$ to be constant before and after partition.

In order to have no worse model convergence bound after partitioning, we need $h(P, G)$ to be non-increasing with the reduction of training resources P. As proposed in Lemma 4.1, Auxo partitions K cohorts when the intra-cohort heterogeneity can be reduced by $\sqrt{K}$ times, which approximately give $\frac{G^2}{P}$ be constant as the one before partition $\frac{G_0^2}{P_0}$. By substituting this relationship into $h(P, G)$, we can derive the lower bound for the range of training resources required to achieve better convergence bound:

$$P \geq \sqrt{\frac{\theta P_0}{G_0^2\rho}} = \sqrt{\frac{\sigma^2}{18k\widetilde{\eta}^2\beta}\frac{P_0}{G_0^2}} = \alpha$$