

# Harnessing Increased Client Participation with Cohort-Parallel Federated Learning

Akash Dhasade  
EPFL  
Lausanne, Switzerland

Anne-Marie Kermarrec  
EPFL  
Lausanne, Switzerland

Tuan-Anh Nguyen  
Independent Researcher  
Grenoble, France

Rafael Pires  
EPFL  
Lausanne, Switzerland

Martijn de Vos  
EPFL  
Lausanne, Switzerland

## Abstract

Federated learning (FL) is a machine learning approach where nodes collaboratively train a global model. As more nodes participate in a round of FL, the effectiveness of individual model updates by nodes also diminishes. In this study, we increase the effectiveness of client updates by dividing the network into smaller partitions, or *cohorts*. We introduce Cohort-Parallel Federated Learning (CPFL): a novel learning approach where each cohort independently trains a global model using FL, until convergence, and the produced models by each cohort are then unified using knowledge distillation. The insight behind CPFL is that smaller, isolated networks converge quicker than in a one-network setting where all nodes participate. Through exhaustive experiments involving realistic traces and non-IID data distributions on the CIFAR-10 and FEMNIST image classification tasks, we investigate the balance between the number of cohorts, model accuracy, training time, and compute resources. Compared to traditional FL, CPFL with four cohorts, non-IID data distribution, and CIFAR-10 yields a 1.9× reduction in train time and a 1.3× reduction in resource usage, with a minimal drop in test accuracy.

**CCS Concepts:** • Computing methodologies → Distributed algorithms.

**Keywords:** Federated Learning, Knowledge Distillation

## ACM Reference Format:

Akash Dhasade, Anne-Marie Kermarrec, Tuan-Anh Nguyen, Rafael Pires, and Martijn de Vos. 2025. Harnessing Increased Client Participation with Cohort-Parallel Federated Learning. In *The 5th Workshop on Machine Learning and Systems (EuroMLSys '25)*, March 30-April 3 2025, Rotterdam, Netherlands. ACM, New York, NY, USA, 12 pages. <https://doi.org/10.1145/3721146.3721939>

## 1 Introduction

Federated learning (FL) allows for the collaborative training of a machine learning model across a distributed network of training nodes, or clients, without ever moving training data [34]. A central server orchestrates the process by selecting a subset of clients, referred to as a *cohort* [9], and sends them the most recent version of the global model. Subsequently, clients in this cohort undertake a few training steps on their local datasets, contributing to the refinement of the model. The locally updated models are then transmitted back to the server for aggregation. This iterative process continues with the server selecting another cohort, possibly composed of a different set of clients, for each successive training round until the global model converges.

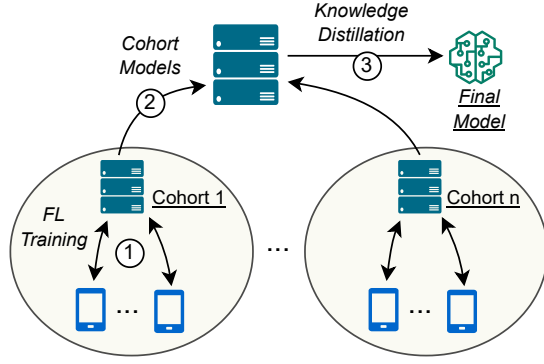
The effect of the cohort size on FL performance has been assessed in numerous studies [9, 10, 22, 34]. While larger cohort sizes intuitively learn from more data in each round, thus accelerating the convergence of the global model [2], they have been found to yield diminishing returns [9]. Furthermore, larger cohorts often use client updates inefficiently, requiring more resources and time to reach similar accuracy levels compared to smaller cohort sizes [9, 22]. As a result, current methods struggle to fully take advantage of model updates from a large number of clients [5].

This work explores a strategy that harnesses increased client participation more efficiently. FL research traditionally focuses on the one-cohort setting and speeds up model convergence by increasing the size of a cohort [5]. We instead propose and investigate the simple idea of *partitioning* the network into several cohorts<sup>1</sup>, each of which runs independent and parallel FL training sessions. We name this approach *Cohort-Parallel Federated Learning*, or CPFL. The

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).  
*EuroMLSys '25, March 30-April 3 2025, Rotterdam, Netherlands*  
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 979-8-4007-1538-9/2025/03  
<https://doi.org/10.1145/3721146.3721939>

<sup>1</sup>We refer to network partitions as *cohorts* throughout this work.



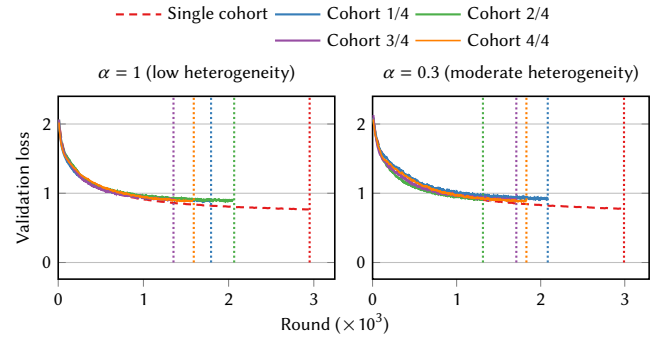
**Figure 1.** The architecture of Cohort-Parallel Federated Learning (CPFL).

architecture of CPFL is illustrated in Figure 1. Clients within a cohort contribute to training an FL model specific to that cohort (step 1), leveraging any existing FL algorithm [1, 27, 34]. This trained cohort model is then uploaded to the global FL server (step 2). Finally, the server unifies these multiple cohort models into a single global model through the process of Knowledge distillation (KD) (step 3). KD is a technique that combines the knowledge of different teacher models into a unified student model [18]. We leverage cross-domain and unlabeled public datasets to carry out KD and produce the final model.

CPFL provides three main benefits compared to single-cohort FL. Firstly, smaller networks make more efficient use of client updates, *reducing the computation and communication resources* required to train a model. Secondly, smaller networks *converge significantly quicker* than the entire network, reducing overall training time. We illustrate this effect in Figure 2, which shows the evolution of validation loss without partitioning the network (dashed curve) and when partitioning the network (solid curves). We annotate with vertical lines the round when models have converged, according to our stopping criterion described in Section 4.1. From this figure, it is evident that smaller partitions converge faster than when the network is not partitioned, both with IID and non-IID data distributions. Thirdly, partitioning the network into smaller cohorts provides a flexible means to control resource usage and time to convergence by appropriately choosing number of cohorts. Our proposal is generic and can therefore be applied to any FL setting.

**Contributions.** This work makes the following contributions:

- We propose CPFL, which uses partitioning as a simple and effective strategy to improve FL efficiency in terms of time-to-accuracy and training resource usage (Section 3).



**Figure 2.** Comparing validation loss for partitioned (solid curves) and unpartitioned (dashed curve) networks across IID (left figure) and non-IID (right figure) distributions. The vertical dotted line denotes the convergence point of the training. Additional details on the experiment setup are provided in Appendix B.1.

- We provide theoretical guarantees on the performance of the global model through domain adaptation analysis. Our result extends KD to a more general setting where the teacher model is composed of a mixture of distributions, as is the case for CPFL (Section 3.2).
- We conduct extensive experiments using realistic traces of devices exhibiting different compute and network speeds on two image classification datasets with varying data distributions (Section 4). We analyze the effect of the number of cohorts in CPFL on the achieved test accuracy, resource utilization, and training time. Our results on the CIFAR-10 dataset under non-IID data distributions demonstrate that employing just four cohorts can already lead to a 1.3 $\times$  reduction in training resource usage and 1.9 $\times$  faster convergence with a minimal drop in test accuracy of 0.6% compared to traditional FL.

In summary, CPFL offers FL practitioners a simple and pragmatic method to obtain considerable resource savings and shorter training sessions in FL systems.

## 2 Related Work

**Cohorts in Federated Learning.** The idea of grouping clients *under some criterion* referred to as *clustering* has been well-studied in the FL literature [14, 15, 32]. The first set of these works leverages clustering to solve the federated multi-task learning (FMTL) problem, which assumes that there exist  $k$  different data distributions  $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_K$  in a network of  $m$  clients. Each distribution  $\mathcal{D}_i$  corresponds to a different task  $i$  where these approaches aim to cluster clients that solve similar tasks without the explicit knowledge of cluster identities [15, 36]. The second set of works focuses on mitigating the impact of non independent and identically distributed (non-IID) data or client data distribution

shift (e.g., concept shift) by clustering statistically similar clients [7, 32]. Such clustering relies on some statistics collected during training e.g., similarity of gradient or model updates obtained from clients, and typically introduces overheads to obtain clusters. Other works use clustering to build a multi-tier hierarchical topology considering both data distribution and communication efficiency [33]. In contrast to the above works, we study and showcase the benefits of *partitioning* i.e., simply dividing the network into arbitrary groups of clients. Uniquely, our work explores the trade-offs between time to convergence, resource usage, and attained accuracies intricately tied to the number of partitions.

**Parallelism in Federated Learning.** The parallelism induced by increasing the number of participating clients per round was initially investigated by McMahan *et al.* [34]. They observed that leveraging more clients per round reduced the number of rounds required to reach a target accuracy. However, the benefits extended only to a certain threshold number of clients, beyond which the returns diminished significantly. Charles *et al.* [9] further examined the impact of large client participation across various learning tasks. The conclusions were similar, where the threshold was empirically shown to lie between 10 and 50, interestingly the same for all tasks. In a network of thousands of clients with several hundred available clients per round, this empirical threshold clearly illustrates the limitations of current methods in effectively utilizing increased client participation. [9] also show that federated algorithms under large client participation use local updates inefficiently, requiring significantly more samples per unit-accuracy. These limitations call for a novel approach that can better capitalize increased client participation [5].

**Knowledge Distillation (KD).** In this work, we leverage KD to combine the knowledge of individual cohort models into a single global model at the server. KD was initially proposed to extract information from a complex teacher model into a small student model [18]. Conventionally, the training of the student model involves the minimization of the disparity between the logits produced by the student and teacher, which are computed utilizing an appropriate auxiliary dataset [18]. KD has been increasingly used in FL to reduce communication costs [16, 17, 35], enable heterogeneous client models [29, 31] or mitigate privacy risks [16]. More generally, KD has also been used to exchange knowledge in a distributed network of clients [4, 38].

**CPFL extremes.** We highlight two algorithms in the literature that are special cases of our proposed CPFL algorithm. One extreme is the one-shot FedKD algorithm [16] wherein each node or client is its own cohort. On the other extreme is the standard FedAvg algorithm [34] where all nodes belong to a single cohort. In particular, FedAvg does not utilize KD since there is only one cohort, requiring no knowledge

---

**Algorithm 1:** Cohort-Parallel Federated Learning
 

---

**Input:** Set of  $n$  cohorts  $\{C_i\}_{i=1}^n$ , aggregation weights  $\{p_i\}_{i=1}^n$ , learning rate  $\eta$ , public dataset  $\hat{\mathcal{D}}_p$ , parametric function  $f$  from  $h_\theta$ , number of epochs  $E$

**Output:** Final global model parameters  $\theta$

**Cohort Servers Execute**

**for** each cohort  $i = 1, \dots, n$  **in parallel** **do**  
      $\theta_i \leftarrow$  train model using standard FL  
     Send  $\theta_i$  to the global server  
**end**

**Global Server Executes**

Initialize global model parameters  $\theta_s$   
**for**  $x \in \hat{\mathcal{D}}_p$  **do**  
     **for**  $i = 1, \dots, n$  **do**  
          $z_i \leftarrow f(x; \theta_i)$   
     **end**  
      $\tilde{z} \leftarrow \sum_{i=1}^n p_i z_i$  ▷ aggregate logits  
**end**  
**for**  $e = 1, \dots, E$  **do**  
     **for** mini-batch  $b \in \hat{\mathcal{D}}_p$  **do**  
          $z_s \leftarrow f(b; \theta_s)$   
          $\theta_s \leftarrow \theta_s - \eta \cdot \nabla_{\theta_s} \mathcal{L}$  ▷ KD using eq. (3)  
     **end**  
**end**  
**return**  $\theta$

---

fusion. CPFL can proficiently navigate this spectrum, exhibiting interesting characteristics that can be controlled by altering the number of cohorts.

### 3 Cohort-Parallel Federated Learning

We now describe CPFL, a learning approach that combines the strength of multiple parallel FL sessions with KD. We illustrate our approach in Figure 1 and provide pseudo code in Algorithm 1.

#### 3.1 Algorithm overview

Consider a supervised learning problem with input space  $\mathcal{X}$  and output space  $\mathcal{Y}$ . For a model  $h_\theta : \mathcal{X} \rightarrow \mathcal{Z}$  parameterized by  $\theta \in \mathbb{R}^d$ , each data point  $(x, y) \in \mathcal{X} \times \mathcal{Y}$  incurs a loss of  $\ell(h_\theta(x), y)$  where  $\ell : \mathcal{Z} \times \mathcal{Y} \rightarrow \mathbb{R}$  is a non-negative loss function. The expected loss of  $h_\theta$  on data distribution  $\mathcal{D}$  is defined as  $\mathcal{L}_{\mathcal{D}}(\theta) = \mathbb{E}_{(x,y) \sim \mathcal{D}} [\ell(h_\theta(x), y)]$ . We consider an FL setting with  $M$  clients and our algorithm proceeds as follows.

In the first stage of the algorithm, the global server randomly partitions the clients into  $n$  cohorts, each comprising  $K$  clients such that  $n * K = M$ .<sup>2</sup> We opt for a random partitioning of nodes into cohorts due to its simplicity and universality. This approach ensures unbiased division without introducing added complexities or biases that more advanced partitioning strategies might incur<sup>3</sup>. Each client has a private local dataset following its local distribution  $\mathcal{D}_{i,k}, \forall i \in [n], \forall k \in [K]$ . Cohorts operate in parallel, running independent FL training sessions using any traditional FL algorithm, such as FedAvg, while reporting to their respective cohort servers, as shown in Step 1 of Figure 1. These servers could, for example, correspond to geographically distributed FL servers running FL sessions within their geographic boundaries or correspond to co-located servers within a single global server. Each such server trains cohort specific parameters  $\theta_i$ , which minimizes the average risk across all clients within the cohort. Precisely, for every  $i \in [n]$ ,

$$\theta_i \in \operatorname{argmin}_{\theta} \frac{1}{K} \sum_{k=1}^K \mathcal{L}_{\mathcal{D}_{i,k}}(\theta). \quad (1)$$

Once training converges, the cohort server transmits  $\theta_i$  to the global server, as shown in Step 2 of Figure 1.

In the second stage of the algorithm, the global server distills the knowledge of each cohort model (called teacher models) into a single global model (called the student model) once all cohorts have converged, shown in step 3 in Figure 1. This knowledge transfer is facilitated by an unlabeled public dataset  $\hat{\mathcal{D}}_p$ . Specifically, the global server generates logit vectors  $\mathbf{z}_i = f(\mathbf{x}; \theta_i)$  for every  $\mathbf{x} \in \hat{\mathcal{D}}_p$  for each cohort model  $i \in [n]$ . These logits are then aggregated  $\tilde{\mathbf{z}} := \sum_{i=1}^n p_i \mathbf{z}_i$  where  $p_i$  denotes the weights of the aggregation with  $\sum_{i=1}^n p_i = 1$  to act as soft-targets for the process of knowledge distillation. Denoting by  $\theta_s$  the parameters of the global server's model and  $\mathbf{z}_s = f(\mathbf{x}; \theta_s)$ , the global server solves the following optimization problem:

$$\theta_s \in \operatorname{argmin}_{\theta} \mathbb{E}_{\mathbf{x} \sim \mathcal{D}_p} [\mathcal{L}(\mathbf{z}_s, \tilde{\mathbf{z}})] \quad (2)$$

$$\text{where } \mathcal{L}(\mathbf{z}_s, \tilde{\mathbf{z}}) = \|\mathbf{z}_s - \tilde{\mathbf{z}}\|_1 \quad (3)$$

and  $\|\cdot\|_1$  represents the L1 norm. Our complete approach is outlined in Algorithm 1.

We set the weights  $\{p_i\}_{i=1}^n$  for logit aggregation by extending the approach used in one-shot FedKD [16] based on the label distribution of nodes. Each cohort server first aggregates the label distributions of its nodes, forming a cohort-wide distribution. The global server then assigns per cohort weight based on its aggregated label distribution. Compared to unweighted averaging, this significantly

boosts the effectiveness of knowledge distillation since particular cohorts might be better suited to predict particular target classes [16]. However, sharing of label distributions by nodes within a cohort may pose privacy risks. To mitigate this risk, one can compute aggregated label distributions using secure hardware [12] or via secure aggregation [6], avoiding the leakage of individual label distributions. Lastly, we highlight that CPFL can leverage any existing FL algorithm in stage 1, ranging from FedAvg [34] to more advanced algorithms [1, 27].

### 3.2 Cross-domain analysis

In line with our setting, we established a bridge between knowledge distillation and the theory of domain adaptation. The concept of domain adaptation revolves around training a classifier to perform effectively in a target domain using a model previously trained in a source domain. This concept mirrors the structure of our framework, where we distill knowledge from models trained in parallel using FL on multiple distinct distributions within a cohort and subsequently transfer it to a global model. By viewing our framework through this lens, we developed a generalization bound for our distilled model by drawing upon the principles of domain adaptation theory [3].

Let  $\mathcal{D}$  represent the target distribution and  $\mathcal{H}$  be the hypothesis class parameterized by  $\Theta \subset \mathbb{R}^d$  as  $\mathcal{H} = \{h_{\theta}, \theta \in \Theta\}$ . With a slight change in notation, we use  $h$  in place of  $h_{\theta}$  for simplicity and indicate by  $\mathcal{L}_{\mathcal{D}}(h)$  the risk of  $h \in \mathcal{H}$ . We define the  $\mathcal{H}\Delta\mathcal{H}$ -divergence between a source distribution  $\mathcal{D}'$  and the target distribution  $\mathcal{D}$ ,  $d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}', \mathcal{D})$  as

$$2 \sup_{h, h' \in \mathcal{H}} |\mathbb{P}_{\mathbf{x} \sim \mathcal{D}'}(h(\mathbf{x}) \neq h'(\mathbf{x})) - \mathbb{P}_{\mathbf{x} \sim \mathcal{D}}(h(\mathbf{x}) \neq h'(\mathbf{x}))|$$

with  $\mathcal{H}\Delta\mathcal{H}$  representing the symmetric difference space defined as  $\mathcal{H}\Delta\mathcal{H} := \{h(\mathbf{x}) \oplus h'(\mathbf{x}) | h, h' \in \mathcal{H}\}$ . Furthermore, we define  $\lambda := \inf_{h \in \mathcal{H}} \{\mathcal{L}_{\mathcal{D}'}(h) + \mathcal{L}_{\mathcal{D}}(h)\}$  as the risk for the optimal hypothesis across the two distributions.

For each cohort  $i \in [n]$ , we designate  $h_i$  and  $\mathcal{D}_i$  as the respective hypothesis and distribution. Given that each cohort conducts federated training in parallel, we characterize the distribution  $\mathcal{D}_i$  as a mixture of distributions from its  $K$  clients i.e.,  $\mathcal{D}_i = \frac{1}{K} \sum_{k=1}^K \mathcal{D}_{i,k}$ . Consequently, we consider a problem of  $n * K$  sources domain adaptation. Denoting the hypothesis on the global server by  $h_s = \sum_{i=1}^n p_i h_i$ , we state the following theorem:

**Theorem 1.** Let  $\mathcal{H}$  be a finite hypothesis class and  $h_s := \sum_{i=1}^n p_i h_i$ , where  $p_i > 0$  and  $\sum_{i=1}^n p_i = 1$ . Suppose that each source dataset has  $m$  instances. Then, for any  $\delta \in (0, 1)$ , with probability at least  $1 - \delta$ , the expected risk of  $h_s$  on the target

<sup>2</sup>This is for simplicity - our setting still holds otherwise.

<sup>3</sup>An advanced partitioning scheme could be a multi-objective function that considers data skewness, device characteristics, etc.

distribution  $\mathcal{D}$  is bounded by:

$$\begin{aligned} \mathcal{L}_{\mathcal{D}}(h_s) &\leq \sum_{i=1}^n \sum_{k=1}^K \frac{p_i}{K} \mathcal{L}_{\hat{\mathcal{D}}_{i,k}}(h_i) \\ &\quad + \sum_{i=1}^n \sum_{k=1}^K \frac{p_i}{2K} \left( \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_{i,k}, \mathcal{D}) + \lambda_{i,k} \right) \quad (4) \\ &\quad + \sqrt{\frac{\log(2nK/\delta)}{2m}} \end{aligned}$$

where  $\lambda_{i,k} := \inf_{h \in \mathcal{H}} \{ \mathcal{L}_{\mathcal{D}_{i,k}}(h) + \mathcal{L}_{\mathcal{D}}(h) \}$  and  $\hat{\mathcal{D}}_{i,k}$  is the observable dataset of the distribution  $\mathcal{D}_{i,k}$ .

We provide a proof in Appendix A. In essence, the theorem establishes an upper bound on the loss of the global model by the weighted sum of: *i)* the risk of each cohort model on its clients' data; *ii)* the discrepancy between the cohort's client distribution and the target distribution, quantified by the second term on the right-hand side; and *iii)* a constant term contingent on the number of data points and the number of sources. Additionally, it highlights the benefits of using multiple small cohorts, which generally achieve better performance on their clients' data compared to larger cohorts. By improving the performance of individual cohort models, we consequently strengthen the performance bound of the global model. Our theorem extends previous works [16, 31] to a more general setting where we distill knowledge from teacher models which themselves are trained on a mixture of their respective client distributions.

## 4 Evaluation

We implement CPFL and explore different trade-offs between achieved accuracy, training time, and resource usage for different data distributions and number of cohorts.

### 4.1 Experiment setup

We have implemented CPFL using PyTorch and published all source code online.<sup>4</sup>

**Dataset.** We experiment with both the CIFAR-10 and FEMNIST datasets. CIFAR-10 is a common and well-known baseline in machine learning and consists of 50 000 color images, divided amongst ten classes [25]. FEMNIST contains 805 263 grayscale images, divided amongst 62 classes [8]. As backbone models, we use a LeNet architecture [28] for CIFAR-10 and a CNN [34] for FEMNIST. We fix the batch size to 20. Clients within a cohort train using the standard FedAvg algorithm utilizing the SGD optimizer with a learning rate of  $\eta = 0.002$  and a momentum of 0.9 for CIFAR-10, and  $\eta = 0.004$  for FEMNIST. These values are taken from existing works [13, 19]. Clients perform one local epoch in a given round for CIFAR-10 and five local steps for FEMNIST.

We consider a network with 200 nodes for CIFAR-10 and take a fixed random subset of 1000 nodes (out of 3550 nodes)

for FEMNIST. We vary the number of cohorts  $n \in [1, 200]$  for CIFAR-10 and  $n \in [1, 64]$  for FEMNIST where for each setting of  $n$ , the nodes are split randomly into  $n$  cohorts. For CIFAR-10, we experiment with varying degrees of non-IIDness, controlled by  $\alpha$ . Specifically, we construct heterogeneous data splits using the Dirichlet distribution, in line with previous work [16, 20, 31]. We experiment with  $\alpha = 1$  (low heterogeneity),  $\alpha = 0.3$  (moderate heterogeneity) and  $\alpha = 0.1$  (high heterogeneity). FEMNIST follows a natural non-IID partitioning based on the data sample creators.

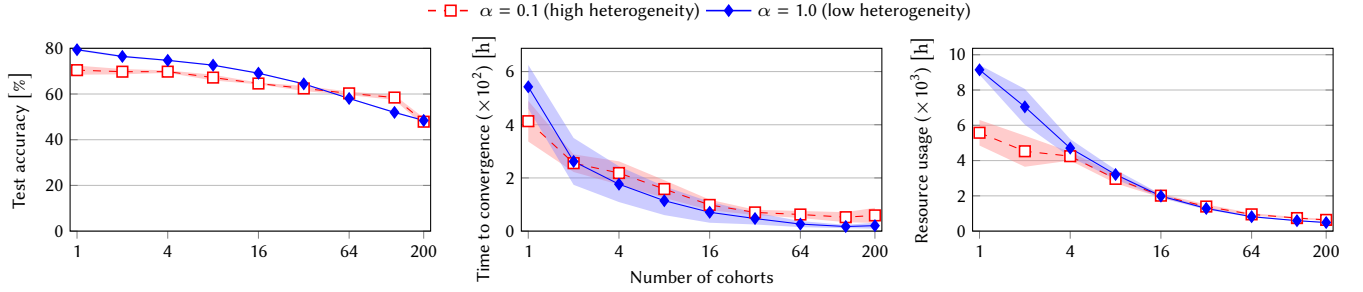
**Distillation.** For CIFAR-10, we perform KD using the STL-10 dataset [11], a dataset inspired by CIFAR-10 that has also been utilized in earlier studies [35]. This dataset contains 100 000 data samples. For FEMNIST, we use the SVHN dataset, containing 531 131 unlabeled images of house numbers [37]. For distillation, we use the Adam optimizer, a learning rate of 0.001, a batch size of 512, and train for 50 epochs.

**Validation set.** To progress to the second stage of the algorithm, CPFL requires a signal to determine when cohorts have finished training. To achieve this, we have nodes within the cohort compute and report the loss of the global model on a local validation set. This validation set is 10% of the local dataset, and only nodes with at least 10 data samples construct this validation set and report their validation loss. The cohort server collects all validation losses during each round and averages them. A cohort stops model training once the minimum validation loss has not decreased for  $r$  rounds. We have conducted various trials, and we found that a value of  $r = 50$  for CIFAR-10 and  $r = 200$  for FEMNIST achieves a reasonable trade-off between waiting too long and letting cohort models sufficiently converge. We also apply a moving average with a window size of 20 to reduce noise.

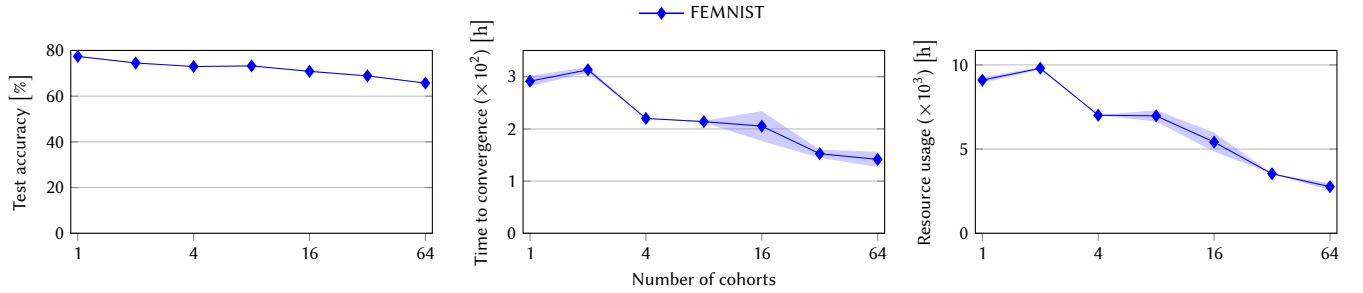
**Traces.** We evaluate CPFL under realistic settings to provide a closer approximation to real-world conditions. To this end, we integrate network and compute capacities traces to simulate the hardware performance of nodes [26]. These traces contain the hardware profile of 131 000 mobile devices and are sourced initially from the AI and MobiPerf benchmarks [21, 23]. They span a broad spectrum ranging from the network speeds of 130 KB/s to 26 MB/s while the compute speeds vary from 0.9 secs to 11.9 secs to train a mini-batch. We assume that the cohort servers have sufficiently large incoming and outgoing bandwidth capacities that can handle all transfers in a round in parallel. Finally, all nodes remain online and available during the experiment.

**Compute infrastructure and implementation.** We run all experiments on machines in our compute cluster. Each machine is equipped with dual 24-core AMD EPYC-2 CPU, 128 GB of memory, a NVIDIA RTX A4000 GPU, and is running CentOS 8. For reproducibility and in line with related work in the domain, we simulate the passing of time in our experiments [1, 26, 27]. We achieve this by customizing

<sup>4</sup>See <https://github.com/sacs-epfl/cpfl>.



**Figure 3.** The test accuracy, convergence time and resource usage (in CPU hours) of CIFAR-10, for increasing number of cohorts ( $n$ ) and different heterogeneity levels (controlled by  $\alpha$ ). Results for  $\alpha = 0.3$  are included in Appendix B.3.



**Figure 4.** The test accuracy, convergence time and resource usage (in CPU hours) of FEMNIST, for increasing number of cohorts ( $n$ ).

the default event loop provided by the `asyncio` library and processing events without delay.

We experiment with FedAvg [34] as the default FL algorithm for training within the cohort. We report top-1 test accuracies when evaluating the final model with the test sets. Lastly, we run each experiment with five different seeds, varying data distribution (for CIFAR-10) and node characteristics, and average all results.

#### 4.2 Time and resource savings by CPFL

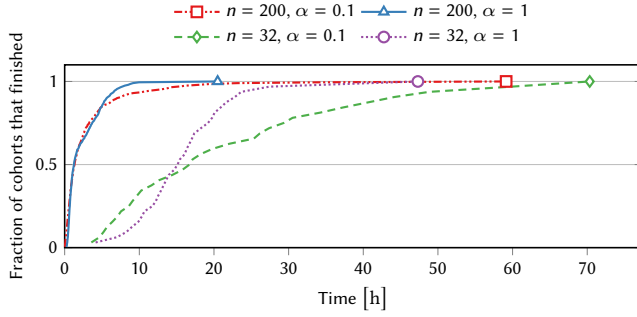
We first evaluate the test accuracy, time to convergence, and resource usage by CPFL, for CIFAR-10 and FEMNIST under different data distributions ( $\alpha$ ) and number of cohorts ( $n$ ). For CIFAR-10, we vary  $n$  from 1 to 200;  $n = 1$  corresponds to the traditional FL setting, whereas  $n = 200$  assigns each node to its own cohort, corresponding to one-shot FedKD as described in Section 2. We remark that one-shot FedKD has only been evaluated with 20 nodes at most, and its performance in larger networks remains unknown. For CIFAR-10 and FEMNIST, we set the client participation rate to 100% and 20% respectively.

**Test accuracy.** Figures 3 and 4 (left plot) show the test accuracy (in %) of CPFL for the CIFAR-10 and FEMNIST datasets respectively, across different values of  $n$ . Both datasets show a decreasing trend in test accuracy as  $n$  increases. This is because (i) each cohort learns on fewer data as  $n$  increases,

and (ii) KD does not perfectly distill all knowledge. For FEMNIST,  $n = 1$  shows a test accuracy of 77.4%, which gradually decreases to 65.7% for  $n = 64$ . For CIFAR-10 with  $\alpha = 0.1$ , accuracy drops from 70.4% ( $n = 1$ ) to 47.9% ( $n = 200$ ). Nonetheless, we observe the drops are marginal when considering lower values of  $n$ . For instance, for CIFAR-10 with  $\alpha = 0.1$ , the accuracy of 70.4% at  $n = 1$  reduces just to 69.8% at  $n = 4$ . Thus lower values of  $n$  make for an interesting case for obtaining significant time and resource savings as we show next.

**Time-to-accuracy.** Figures 3 and 4 (middle plot) show the time to convergence of CPFL for the same evaluated values of  $n$ . This is the time between starting model training and training completion by all cohorts. For CIFAR-10, we notice a stark decrease in time to convergence as  $n$  increases: for  $\alpha = 0.1$  and when increasing  $n$  from 1 to 4, the time until convergence decreases from 413 h to 218 h, a speedup of 1.9 $\times$  with a minimal loss in test accuracy. Similarly, on the FEMNIST dataset, the time until convergence reduces from 290 h ( $n = 1$ ) to 220 h ( $n = 4$ ), with a speedup of 1.3 $\times$ . This speedup is because cohorts with fewer data samples require less time to converge. We remark that Figure 3 excludes the time to complete the KD, but we found this to be minimal compared to the overall training time. We provide more details on this in Appendix B.2. For CIFAR-10 with non-IID data ( $\alpha = 0.1$ ), CPFL can obtain speedups between 1.6–7 $\times$  as  $n$  varies from 2 to 200. These speedups might be particularly





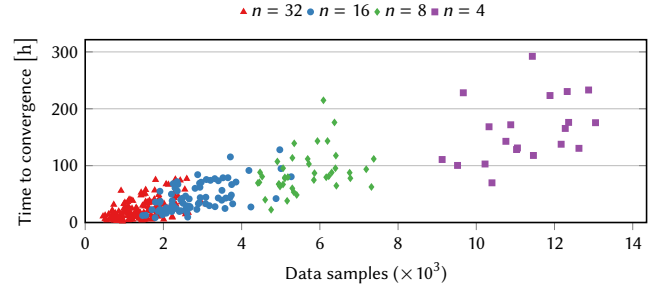
**Figure 5.** The finish times of individual cohorts, for different numbers of total cohorts and data distributions. We mark the finish time of each group with a symbol.

beneficial in practical FL settings where one has to execute many runs, *e.g.*, for hyperparameter tuning [24].

We further analyze the training speedups by CPFL against FL by distilling the models produced by each cohort after a fixed duration. Specifically, we distill 10 and 15 hours into the experiment for  $\alpha = 0.1$  and the CIFAR-10 dataset. 10 and 15 hours into the experiment, FL attained 28.7% and 35.4% test accuracy, respectively. For  $n = 2$ , these accuracies are 36.16% and 39.56%, respectively. For  $n = 16$  the increase in accuracy over FL is even more pronounced, reaching 54.1% and 56.7% accuracy 10 and 15 hours into the experiment, respectively. We attribute these speedups primarily to the decrease in round durations. As  $n$  increases, cohorts become smaller and thus a round requires fewer client updates compared to the setting with  $n = 1$  where the server needs all 200 model updates to conclude a round.

**Resource usage.** Figures 3 and 4 (right plot) visualize the resource usage of CPFL in CPU hours, representing the total time CPUs spent on training. We observe a reduction in resource usage thanks to the faster convergence of cohort models. For CIFAR-10 in the high heterogeneity setting ( $\alpha = 0.1$ ), increasing  $n$  from 1 to 4 decreases the required CPU hours from 5577 h to 4249 h, or 1.3 $\times$ . Similarly for the FEMNIST dataset, the CPU hours reduce from 9093 h ( $n = 1$ ) to 7011 h ( $n = 4$ ), or 1.3 $\times$ . On the other hand, when increasing  $n$  up to 200, a substantial reduction of 8.8 $\times$  is achieved for  $\alpha = 0.1$  for CIFAR-10, albeit with a trade-off of decreased accuracy. Consequently, in the non-IID case with  $\alpha = 0.1$ , CPFL demonstrates the potential to reduce resource usage by 1.2 – 8.8 $\times$  as  $n$  varies from 2 to 200. We also observed reductions in communication volume, both for CIFAR-10 and FEMNIST. They follow a similar trend as the reduction in resource usage, and we further comment on this in Appendix B.4.

**Experimental conclusion.** Our results suggest that a reasonable value for the number of cohorts lies between 4 and 16. We conclude that while FL ( $n = 1$ ) trains more accurate models compared to CPFL, it incurs longer training



**Figure 6.** The relation between the number of data samples and the time until convergence within a cohort, for  $n = 4, 8, 16$  and 32, and under a non-IID data distribution ( $\alpha = 0.1$ ). We have  $5 * n$  measurements for each  $n$  due to 5 seeds.

times and more resource usage. On the other hand, one-shot FedKD ( $n = M$ ) suffers from accuracy loss but shows great potential to reduce training time and resource usage. CPFL can proficiently navigate these two extremes by controlling the number of cohorts, offering FL practitioners a simple way to tailor FL training sessions according to their requirements.

The remaining experiments in this section use the CIFAR-10 dataset as it allows experimentation with different degrees of non-IIDness.

### 4.3 Training time of cohorts

We further analyze the convergence times of cohorts in some of the experiments in Section 4.2. We plot in Figure 5, for  $n = 32$  and  $n = 200$ , and  $\alpha = 1$  and  $\alpha = 0.1$ , an ECDF with the fraction of cohorts that completed training as the experiment progressed. These numbers correspond to the experiment described in Section 4.2 on the CIFAR-10 dataset. In the plot, we annotate the completion of the last cohort in each setting with a marker.

Our first observation is that for  $n = 200$ , 75% of the cohorts converge within less than 5 hours. We do notice some slower cohorts that prolong overall training. Compared to when using  $\alpha = 1$ , this slowdown is more pronounced for  $\alpha = 0.1$ , where data distribution is much more skewed. We also notice similar effects for  $n = 32$ , and we observe a higher variation of training times across cohorts for  $n = 32$  and  $\alpha = 0.1$ , compared to  $n = 32$  and  $\alpha = 1$ . Our results in Figure 5 suggest that FL practitioners can further gain speedups by proceeding to the KD step even when a fraction of cohorts (*e.g.*, 75%) have converged instead of waiting for all cohorts to finish training. This is similar in spirit to how the federated server in traditional FL tolerates partial updates when not all clients complete the expected number of local steps in specified time [30], albeit at the cost of accuracy.

### 4.4 Cohort data samples and training time

CPFL is based on the premise that smaller cohorts *i.e.*, cohorts with fewer clients and consequently with fewer total data

$\alpha$	$n$	Teacher acc.	Student acc.	$\Delta$
0.1	4	65.76 $\pm$ 2.63	69.79 $\pm$ 0.55	+4.03
	16	48.24 $\pm$ 5.86	64.59 $\pm$ 0.39	+16.35
	64	27.64 $\pm$ 7.38	60.28 $\pm$ 0.80	+32.65
	200	17.32 $\pm$ 5.67	47.89 $\pm$ 1.75	+30.57
1	4	74.44 $\pm$ 0.86	74.74 $\pm$ 0.30	+0.30
	16	63.37 $\pm$ 1.76	69.11 $\pm$ 0.34	+5.74
	64	42.44 $\pm$ 5.17	58.05 $\pm$ 0.24	+15.61
	200	27.00 $\pm$ 5.29	48.43 $\pm$ 0.27	+21.44

**Table 1.** The average accuracy of teacher and student models for varying values of  $\alpha$  and  $n$ , and for the CIFAR-10 dataset. The right-most column shows the improvement in test accuracy by knowledge distillation.

samples, converge quicker. To validate this premise, we plot in Figure 6 the relation between the number of total data samples and the time until convergence for every cohort under a non-IID data distribution ( $\alpha = 0.1$ ). We extract results for  $n = 4, 8, 16$  and  $32$  for the experiment runs described in Section 4.2. Figure 6 hints at a positive relation between the number of data samples in a cohort and the time until convergence. Therefore, FL practitioners can manipulate  $n$  to increase or decrease the average number of data samples per cohort, subsequently controlling the time required for the cohort to finish training.

#### 4.5 Teacher and student accuracies of KD

We now assess the accuracy improvement by KD. We show in Table 1 the average test accuracy of the teacher and student models and the average improvement ( $\Delta$ ), for  $\alpha = 0.1$  and  $\alpha = 1$  across four different values of  $n$ . We also report the corresponding standard deviations.

Table 1 shows that increasing  $n$  improves the gain in test accuracy induced by KD. This is because, with higher values of  $n$ , each cohort trains on proportionately less data, affecting the generalization performance of the teacher models. KD in this case draws a larger *relative* improvement. We also observe that the gains in accuracy are higher for  $\alpha = 0.1$  compared to  $\alpha = 1.0$ . We suspect this is because under high heterogeneity, KD is more capable of combining models that recognize specialized features compared to low heterogeneity settings where mostly similar models are produced. This observation also aligns with KD’s original concept as a method to effectively combine knowledge from *specialized* ensembles, shown in [18].

## 5 Conclusion

This paper introduced Cohort-Parallel Federated Learning (CPFL), an innovative approach to enhance FL by partitioning the network into several smaller cohorts. The underpinning principle of CPFL is that smaller networks lead to

quicker convergence and more efficient resource utilization. By unifying the cohort models with Knowledge Distillation and a cross-domain, unlabeled dataset, we produce a final global model that integrates the contributions from all clients. Our experimental findings confirm that this strategy yields significant advantages in training time and resource utilization without considerably compromising test accuracy. This approach offers practitioners a tangible means to control their FL resource consumption and convergence timelines.

## Acknowledgments

This work has been funded by the Swiss National Science Foundation, under the project “FRIDAY: Frugal, Privacy-Aware and Practical Decentralized Learning”, SNSF proposal No. 10.001.796.

## References

- [1] Ahmed M. Abdelmoniem, Atal Narayan Sahu, Marco Canini, and Suhaib A. Fahmy. Refl: Resource-efficient federated learning. In *Proceedings of the Eighteenth European Conference on Computer Systems*, EuroSys '23, page 215–232, New York, NY, USA, 2023. Association for Computing Machinery.
- [2] Sheikh Shams Azam, Martin Pelikan, Vitaly Feldman, Kunal Talwar, Jan Silovsky, and Tatiana Likhomanenko. Federated learning for speech recognition: Revisiting current trends towards large-scale asr. In *International Workshop on Federated Learning in the Age of Foundation Models in Conjunction with NeurIPS 2023*, 2023.
- [3] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79:151–175, 2010.
- [4] Ilai Bistriz, Ariana Mann, and Nicholas Bambos. Distributed distillation for on-device learning. *Advances in Neural Information Processing Systems*, 33:22593–22604, 2020.
- [5] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečný, Stefano Mazzocchi, Brendan McMahan, et al. Towards federated learning at scale: System design. *Proceedings of machine learning and systems*, 1:374–388, 2019.
- [6] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 1175–1191, New York, NY, USA, 2017. Association for Computing Machinery.
- [7] Christopher Briggs, Zhong Fan, and Peter Andras. Federated learning with hierarchical clustering of local updates to improve training on non-iid data. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9, 2020.
- [8] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečný, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. In *2nd Intl. Workshop on Federated Learning for Data Privacy and Confidentiality (FL-NeurIPS)*, 2019.
- [9] Zachary Charles, Zachary Garrett, Zhouyuan Huo, Sergei Shmulyan, and Virginia Smith. On large-cohort training for federated learning. *Advances in neural information processing systems*, 34:20461–20475, 2021.
- [10] Zachary Charles, Nicole Mitchell, Krishna Pillutla, Michael Reneer, and Zachary Garrett. Towards federated foundation models: Scalable dataset pipelines for group-structured learning. *Advances in Neural Information Processing Systems*, 36, 2024.



- [11] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223. JMLR Workshop and Conference Proceedings, 2011.
- [12] Akash Dhasade, Nevena Dresevic, Anne-Marie Kermarrec, and Rafael Pires. Tee-based decentralized recommender systems: The raw data sharing redemption. In *2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 447–458. IEEE, 2022.
- [13] Akash Dhasade, Anne-Marie Kermarrec, Rafael Pires, Rishi Sharma, and Milos Vujasinovic. Decentralized learning made easy with decentralizepy. In *Proceedings of the 3rd Workshop on Machine Learning and Systems*, pages 34–41, 2023.
- [14] Moming Duan, Duo Liu, Xinyuan Ji, Yu Wu, Liang Liang, Xianzhang Chen, Yujuan Tan, and Ao Ren. Flexible clustered federated learning for client-level data distribution shift. *IEEE Transactions on Parallel and Distributed Systems*, 33(11):2661–2674, 2021.
- [15] 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.
- [16] Xuan Gong, Abhishek Sharma, Srikrishna Karanam, Ziyang Wu, Terence Chen, David Doermann, and Arun Innanje. Preserving privacy in federated learning with ensemble cross-domain knowledge distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11891–11899, 2022.
- [17] Chaoyang He, Murali Annavaram, and Salman Avestimehr. Group knowledge transfer: Federated learning of large cnns at the edge. *Advances in Neural Information Processing Systems*, 33:14068–14080, 2020.
- [18] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.
- [19] Kevin Hsieh, Amar Phanishayee, Onur Mutlu, and Phillip Gibbons. The non-iid data quagmire of decentralized machine learning. In *International Conference on Machine Learning*, pages 4387–4398. PMLR, 2020.
- [20] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.
- [21] Junxian Huang, Cheng Chen, Yutong Pei, Zhaoguang Wang, Zhiyun Qian, Feng Qian, Birjodh Tiwana, Qiang Xu, Z Mao, Ming Zhang, et al. Mobiperf: Mobile network measurement system. *Technical Report. University of Michigan and Microsoft Research*, 2011.
- [22] Dzmitry Huba, John Nguyen, Kshitiz Malik, Ruiyu Zhu, Mike Rabbat, Ashkan Yousefpour, Carole-Jean Wu, Hongyuan Zhan, Pavel Ustinov, Harish Srinivas, et al. Papaya: Practical, private, and scalable federated learning. *Proceedings of Machine Learning and Systems*, 4:814–832, 2022.
- [23] Andrey Ignatov, Radu Timofte, Andrei Kulik, Seungsoo Yang, Ke Wang, Felix Baum, Max Wu, Lirong Xu, and Luc Van Gool. Ai benchmark: All about deep learning on smartphones in 2019. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 3617–3635. IEEE, 2019.
- [24] Mikhail Khodak, Renbo Tu, Tian Li, Liam Li, Maria-Florina F Balcan, Virginia Smith, and Ameet Talwalkar. Federated hyperparameter tuning: Challenges, baselines, and connections to weight-sharing. *Advances in Neural Information Processing Systems*, 34:19184–19197, 2021.
- [25] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images, 2009.
- [26] Fan Lai, Yinwei Dai, Sanjay Singapuram, Jiachen Liu, Xiangfeng Zhu, Harsha Madhyastha, and Mosharaf Chowdhury. FedScale: Benchmarking model and system performance of federated learning at scale. In *International Conference on Machine Learning*, pages 11814–11827. PMLR, 2022.
- [27] Fan Lai, Xiangfeng Zhu, Harsha V. Madhyastha, and Mosharaf Chowdhury. Oort: Efficient federated learning via guided participant selection. In *15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21)*, pages 19–35. USENIX Association, July 2021.
- [28] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, Richard Howard, Wayne Hubbard, and Lawrence Jackel. Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, 2, 1989.
- [29] Daliang Li and Junpu Wang. Fedmd: Heterogeneous federated learning via model distillation. *NeurIPS Workshop on Federated Learning for Data Privacy and Confidentiality*, 2019.
- [30] 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.
- [31] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 2351–2363. Curran Associates, Inc., 2020.
- [32] Jiachen Liu, Fan Lai, Yinwei Dai, Aditya Akella, Harsha V. Madhyastha, and Mosharaf Chowdhury. Auxo: Efficient federated learning via scalable client clustering. In *Proceedings of the 2023 ACM Symposium on Cloud Computing*, SoCC '23, page 125–141, New York, NY, USA, 2023. Association for Computing Machinery.
- [33] Qianpiao Ma, Jianchun Liu, Hongli Xu, Qingmin Jia, and Renchao Xie. Fractal: Data-aware clustering and communication optimization for decentralized federated learning. *IEEE Transactions on Big Data*, 2024.
- [34] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.
- [35] Felix Sattler, Arturo Marban, Roman Rischke, and Wojciech Samek. Cfd: Communication-efficient federated distillation via soft-label quantization and delta coding. *IEEE Transactions on Network Science and Engineering*, 9(4):2025–2038, 2022.
- [36] 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.
- [37] Pierre Sermanet, Soumith Chintala, and Yann LeCun. Convolutional neural networks applied to house numbers digit classification. In *Proceedings of the 21st international conference on pattern recognition (ICPR2012)*, pages 3288–3291. IEEE, 2012.
- [38] Andrey Zhmoginov, Mark Sandler, Nolan Miller, Gus Kristiansen, and Max Vladymyrov. Decentralized learning with multi-headed distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8053–8063, 2023.

## A Theoretical Analysis

We begin by recalling the notations and definitions used throughout the theoretical analysis. We define  $\mathcal{H}\Delta\mathcal{H}$ -divergence between a source  $\mathcal{D}'$  and target  $\mathcal{D}$  distribution as:

$$d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}', \mathcal{D}) = 2 \sup_{h, h' \in \mathcal{H}} \left| \mathbb{P}_{x \sim \mathcal{D}'}(h(x) \neq h'(x)) - \mathbb{P}_{x \sim \mathcal{D}}(h(x) \neq h'(x)) \right|$$

Let  $h_s = \sum_{i=1}^n p_i h_i$  where  $\sum_{i=1}^n p_i = 1$  and  $p_i > 0$ , denote the hypothesis on the global server and  $h_i$  the one on the cohort  $i$ . Let  $\mathcal{D}$  be any target distribution and  $\mathcal{D}_i$  be a mixture of  $K$ 's local distribution  $\{\mathcal{D}_{i,k}\}_{k=1}^K$ , that is,  $\mathcal{D}_i = \frac{1}{K} \sum_{k=1}^K \mathcal{D}_{i,k}$ . Denote the expected risk of hypothesis  $h$  on the distribution  $\mathcal{D}$  as  $\mathcal{L}_{\mathcal{D}}(h)$ . We state the following lemmas.

*Lemma 2* (Theorem 3, [3]). For any target distribution  $\mathcal{D}$  and source distribution  $\mathcal{D}'$ . The expected risk of a hypothesis  $h$  on the  $\mathcal{D}$  is bounded by

$$\mathcal{L}_{\mathcal{D}}(h) \leq \mathcal{L}_{\mathcal{D}'}(h) + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}', \mathcal{D}) + \lambda \quad (5)$$

where we note  $\lambda = \inf_{h \in \mathcal{H}} \{\mathcal{L}_{\mathcal{D}'}(h) + \mathcal{L}_{\mathcal{D}}(h)\}$

*Lemma 3.* Let  $\{\mathcal{D}_k\}_{k=1}^K$  be a set of distributions and define  $\mathcal{D}' = \frac{1}{K} \sum_{k=1}^K \mathcal{D}_k$ . For any distribution  $\mathcal{D}$ , the  $\mathcal{H}\Delta\mathcal{H}$ -divergence between  $\mathcal{D}'$  and  $\mathcal{D}$  is bounded by

$$d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}', \mathcal{D}) \leq \frac{1}{K} \sum_{k=1}^K d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_k, \mathcal{D}) \quad (6)$$

*Proof.*

$$\begin{aligned} & d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}', \mathcal{D}) \quad (7) \\ &= 2 \sup_{h, h' \in \mathcal{H}} \left| \mathbb{P}_{x \sim \mathcal{D}'}(h(x) \neq h'(x)) - \mathbb{P}_{x \sim \mathcal{D}}(h(x) \neq h'(x)) \right| \\ &= 2 \sup_{h, h' \in \mathcal{H}} \left| \mathbb{P}_{x \sim \frac{1}{K} \sum_{k=1}^K \mathcal{D}_k}(h(x) \neq h'(x)) - \mathbb{P}_{x \sim \mathcal{D}}(h(x) \neq h'(x)) \right| \\ &= 2 \sup_{h, h' \in \mathcal{H}} \left| \frac{1}{K} \sum_{k=1}^K \mathbb{P}_{x \sim \mathcal{D}_k}(h(x) \neq h'(x)) - \mathbb{P}_{x \sim \mathcal{D}}(h(x) \neq h'(x)) \right| \\ &\leq \frac{2}{K} \sum_{k=1}^K \sup_{h, h' \in \mathcal{H}} \left| \mathbb{P}_{x \sim \mathcal{D}_k}(h(x) \neq h'(x)) - \mathbb{P}_{x \sim \mathcal{D}}(h(x) \neq h'(x)) \right| \\ &= \frac{1}{K} \sum_{k=1}^K d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_k, \mathcal{D}) \quad (8) \end{aligned}$$

□

*Theorem 1.* Let  $\mathcal{H}$  be a finite hypothesis class and  $h_s := \sum_{i=1}^n p_i h_i$ , where  $p_i > 0$  and  $\sum_{i=1}^n p_i = 1$ . Suppose that each source dataset has  $m$  instances. Then, for any  $\delta \in (0, 1)$ , with probability at least  $1 - \delta$ , the expected risk of  $h_s$  on the target

distribution  $\mathcal{D}$  is bounded by:

$$\begin{aligned} \mathcal{L}_{\mathcal{D}}(h_s) &\leq \sum_{i=1}^n \sum_{k=1}^K \frac{p_i}{K} \mathcal{L}_{\hat{\mathcal{D}}_{i,k}}(h_i) \\ &\quad + \sum_{i=1}^n \sum_{k=1}^K \frac{p_i}{K} \left( \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_{i,k}, \mathcal{D}) + \lambda_{i,k} \right) \\ &\quad + \sqrt{\frac{\log(2nK/\delta)}{2m}} \quad (9) \end{aligned}$$

*Proof.* By the definition of  $h_s$  and the Jensen inequality, we have:

$$\mathcal{L}_{\mathcal{D}}(h_s) := \mathcal{L}_{\mathcal{D}} \left( \sum_{i=1}^n p_i h_i \right) \leq \sum_{i=1}^n p_i \mathcal{L}_{\mathcal{D}}(h_i) \quad (10)$$

Using Lemma 2, the expected risk of  $h_i$  on  $\mathcal{D}$  is bounded by:

$$\mathcal{L}_{\mathcal{D}}(h_i) \leq \mathcal{L}_{\mathcal{D}_i}(h_i) + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_i, \mathcal{D}) + \lambda_i \quad (11)$$

where  $\lambda_i = \inf_{h \in \mathcal{H}} \{\mathcal{L}_{\mathcal{D}}(h) + \mathcal{L}_{\mathcal{D}_i}(h)\}$ . Let  $\mathcal{D}_i = \frac{1}{K} \sum_{k=1}^K \mathcal{D}_{i,k}$  be a mixture of distributions in cohort  $i$ , we have then:

$$\begin{aligned} \mathcal{L}_{\mathcal{D}}(h_i) &\leq \mathcal{L}_{\mathcal{D}_i}(h_i) + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_i, \mathcal{D}) + \lambda_i \\ &\leq \mathcal{L}_{\mathcal{D}_i}(h_i) + \frac{1}{2K} \sum_{k=1}^K d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_{i,k}, \mathcal{D}) + \lambda_i \\ &\leq \frac{1}{K} \sum_{k=1}^K \mathcal{L}_{\mathcal{D}_{i,k}}(h_i) + \frac{1}{2K} \sum_{k=1}^K d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_{i,k}, \mathcal{D}) + \lambda_i \\ &\leq \frac{1}{K} \sum_{k=1}^K \mathcal{L}_{\mathcal{D}_{i,k}}(h_i) + \frac{1}{2K} \sum_{k=1}^K d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_{i,k}, \mathcal{D}) + \frac{1}{K} \sum_{k=1}^K \lambda_{i,k} \quad (12) \end{aligned}$$

where the second inequality is application of Lemma 3. The third one follows the fact that  $\mathcal{D}_i = \frac{1}{K} \sum_{k=1}^K \mathcal{D}_{i,k}$ . The last inequality follows the same fact and application of triangle and Jensen's inequality on  $\lambda_i$ . Using Hoeffding's inequality, with probability  $1 - \frac{\delta}{nK}$ , the risk of hypothesis  $h_i$  on the source distribution  $\mathcal{D}_{i,k}$  is upper bounded by

$$\mathcal{L}_{\mathcal{D}_{i,k}}(h_i) \leq \mathcal{L}_{\hat{\mathcal{D}}_{i,k}}(h_i) + \sqrt{\frac{\log(2nK/\delta)}{2m}} \quad (13)$$

where  $\mathcal{L}_{\hat{\mathcal{D}}_{i,k}}(h_i)$  is the risk of  $h_i$  on the empirical distribution  $\hat{\mathcal{D}}_{i,k}$ . Combining Equations (10), (12) and (13) and using the same analysis as in [31]. With probability at least  $1 - \delta$  over  $nK$  sources of  $m$  samples, we have :

$$\begin{aligned} & \mathcal{L}_{\mathcal{D}}(h_s) \\ &\leq \sum_{i=1}^n \sum_{k=1}^K \frac{p_i}{K} \mathcal{L}_{\mathcal{D}_{i,k}}(h_i) + \sum_{i=1}^n \sum_{k=1}^K \frac{p_i}{2K} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_{i,k}, \mathcal{D}) \\ &\quad + \sum_{i=1}^n \sum_{k=1}^K \frac{p_i}{K} \lambda_{i,k} \end{aligned}$$

$$\begin{aligned}
 &\leq \sum_{i=1}^n \sum_{k=1}^K \frac{p_i}{K} \mathcal{L}_{\hat{\mathcal{D}}_{i,k}}(h_i) + \sum_{i=1}^n \sum_{k=1}^K \frac{p_i}{K} \sqrt{\frac{\log(2nK/\delta)}{2m}} \\
 &\quad + \sum_{i=1}^n \sum_{k=1}^K \frac{p_i}{2K} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_{i,k}, \mathcal{D}) + \sum_{i=1}^n \sum_{k=1}^K \frac{p_i}{K} \lambda_{i,k} \\
 &\leq \sum_{i=1}^n \sum_{k=1}^K \frac{p_i}{K} \mathcal{L}_{\hat{\mathcal{D}}_{i,k}}(h_i) + \sum_{i=1}^n \sum_{k=1}^K \frac{p_i}{K} \left( \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_{i,k}, \mathcal{D}) + \lambda_{i,k} \right) \\
 &\quad + \sqrt{\frac{\log(2nK/\delta)}{2m}} \tag{14}
 \end{aligned}$$

□

## B Additional Notes on Experimental Evaluation

### B.1 Motivational Plot (Figure 2)

Figure 2, showing the evolution of validation loss and motivating the approach behind CPFL, has been generated from the CIFAR-10 experiments described in Section 4.2. These plots correspond to the validation loss for each of the four cohorts, in a setting with  $n = 4$  and when using 90 as seed, for  $\alpha = 1$  and  $\alpha = 0.3$ .

In Figure 2, we chose to highlight the results of a single run rather than averaging across multiple runs. The rationale behind this decision stems from the fact that when averaging, the stopping criterion which applies to individual runs does not hold any meaningful interpretation for the averaged result. This means that an average might misrepresent the number of rounds required for convergence or even the validity of the stopping criterion itself. By focusing on a single run, we ensure the integrity and applicability of our stopping criterion, providing a clearer and more direct interpretation of our results. We have manually verified that our conclusion also hold true for different seeds and number of cohorts.

### B.2 Knowledge Distillation

Figure 3 excludes the time it takes for the KD process to complete. For CIFAR-10, we noticed that KD takes between 50 minutes (for  $n = 2$ ) and 305 minutes (for  $n = 200$ ) to complete. This time frame represents a small fraction of the time and resources required for model training by cohorts. Regarding FEMNIST, the KD process takes between 59 minutes (for  $n = 2$ ) and 16.8 hours (for  $n = 64$ ). In the case of FEMNIST, the majority of time is spent on generating inferences from the teacher models. We propose two methods to expedite this process. First, one can parallelize this process by generating logits from distinct teacher models simultaneously. Second, one can increase the inference batch size, although at the cost of additional memory usage.

### B.3 CIFAR-10 Results with $\alpha = 0.3$

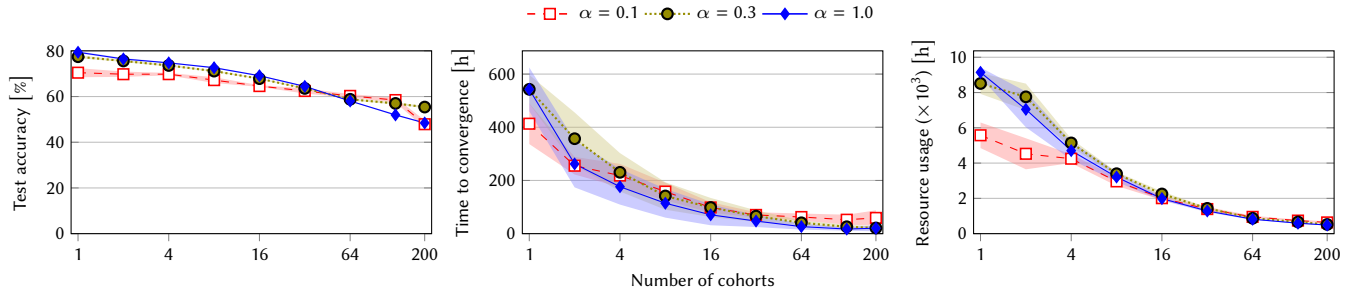
We show in Figure 7 the test accuracy, convergence time and resource usage (CPU hours) of the CIFAR-10 dataset.

Compared to Figure 3, this plot includes the  $\alpha = 0.3$  setting that we omitted from Figure 3 for presentation clarity.

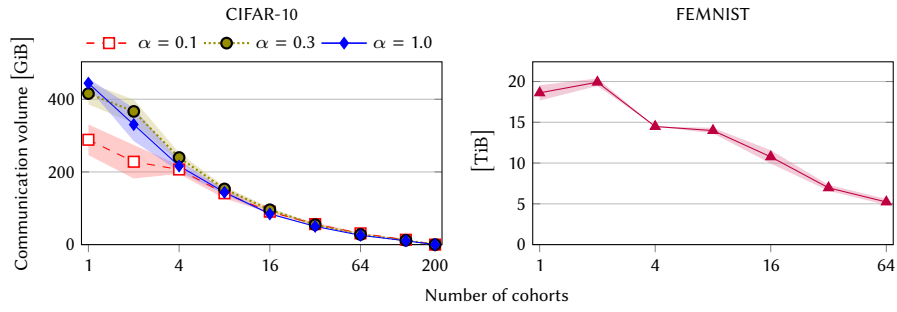
### B.4 Savings in Communication Volume by CPFL

Besides savings in training time and CPU resource usage (see Figure 3 and 4), CPFL also provides savings in communication volume. Figure 8 shows the communication volume required by CPFL, for the CIFAR-10 and FEMNIST dataset, and for different number of cohorts ( $n$ ). The trend in communication volume as  $n$  increases is comparable to the trend in time to convergence and resource usage shown in Figures 3 and 4. For CIFAR-10 with  $n = 200$ , since each node will perform standalone training of its model, we remark that the only communication volume incurred is when sending the trained cohort model to the global server.

For the same value of  $n$ , we notice that FEMNIST incurs significantly more communication volume compared to CIFAR-10. This is because the model size of FEMNIST in serialized form is significantly larger than that of CIFAR-10, namely 6.7 MB for FEMNIST compared to 346 KB for CIFAR-10. Additionally, FEMNIST also requires many more rounds to converge than CIFAR-10 due to the large overall network size (1000 nodes) and the challenging nature of the 62-class classification task. As a result, this leads to escalated communication costs.



**Figure 7.** The test accuracy, convergence time and resource usage (in CPU hours) of CIFAR-10, for different number of cohorts ( $n$ ) and different heterogeneity levels (controlled by  $\alpha$ ).



**Figure 8.** The communication volume required by CPFL for convergence, for CIFAR-10 and FEMNIST, and for different number of cohorts ( $n$ ). For CIFAR-10 we also show the communication volume for different values of  $\alpha$ .