Federated Learning with Taskonomy for Non-IID Data

Hadi Jamali-Rad[1, 3] Mohammad Abdizadeh[2] Attila Szabó[1]
[1]Shell Global Solutions International B.V., Amsterdam, The Netherlands
[2]Myant Inc., Toronto, ON, Canada
[3]Delft University of Technology (TU Delft), Delft, The Netherlands
{hadi.jamali-rad,attila.szabo}@shell.com,
mohammad.abdizadeh@myant.ca, h.jamalirad@tudelft.nl

Abstract

Classical federated learning approaches incur significant performance degradation in the presence of non-IID client data. A possible direction to address this issue is forming clusters of clients with roughly IID data. Most solutions following this direction are iterative and relatively slow, also prone to convergence issues in discovering underlying cluster formations. We introduce federated learning with taskonomy (FLT) that generalizes this direction by learning the task-relatedness between clients for more efficient federated aggregation of heterogeneous data. In a one-off process, the server provides the clients with a pretrained (and fine-tunable) encoder to compress their data into a latent representation, and transmit the signature of their data back to the server. The server then learns the task-relatedness among clients via manifold learning, and performs a generalization of federated averaging. FLT can flexibly handle a generic client relatedness graph, when there are no explicit clusters of clients, as well as efficiently decompose it into (disjoint) clusters for clustered federated learning. We demonstrate that FLT not only outperforms the existing state-of-the-art baselines in non-IID scenarios but also offers improved fairness across clients.[1]

1 Introduction

Federated learning is a new paradigm that offers significant potential in elevating edge-computing capabilities in modern massive distributed networks. While presenting great potential, federated learning also comes with its own unique challenges in practical settings [1, 2]. Recent studies focus on systemic heterogeneity [3], communication efficiency [4, 2, 5], privacy concerns [6, 7] and more recently on fairness [8, 9] and robustness across the network of clients [10, 11]. A defining characteristic of massive decentralized networks is stochastic heterogeneity of client data; i.e., clients possess non-independent and identically distributed (non-IID) data. [12] identifies statistical heterogeneity as the root cause for tension between fairness and robustness constraints in federated optimization. [4, 13] investigate the impact of heterogeneous data distributions on the performance of federated averaging algorithm, FedAvg [4], and demonstrate significant performance degradation in non-IID settings. Personalized federated learning tackles data heterogeneity by forming personalized models for clients via meta-learning or multi-task learning [14, 15, 16, 12]. Clustered federated learning addresses this problem by iterative (or recursive) assignment of clients to separate clusters based on model or model update comparisons at the server side [17, 18, 19, 20, 21]. The effectiveness of clustering approaches hinges upon the quality of cluster formation through this iterative assignment process. Besides, clustered federated learning approaches are sensitive to initialization, as we will demonstrate later on.

Inspired by the idea of "taskonomy" [22], we explore the task-relatedness across client data distributions and cast it in the form of a client relatedness graph. This is accomplished in a *one-off* fashion based on contractive encoding of client data followed by manifold learning (with UMAP) at the server side. The proposed approach (FLT) can flexibly

[1]Our codebase can be found at: https://github.com/hjraad/FLT/.

handle a range of possibilities in incorporating this client relatedness graph for federated averaging. It can be used in generic form as an extension of `FedAvg` for non-IID data when there are no explicit clusters of clients with similar data distributions, or if need be, can be decomposed with hierarchical clustering (`HC`) to disjoint clusters and transform into clustered federated learning. Our main contributions can be summarized as follows: i) we propose federated learning with taskonomy (`FLT`), which learns the task-relatedness among clients and uses it at the server-side for federated averaging of non-IID data, without requiring any prior knowledge about the data distribution correlations among clients or the number of clusters they belong to; ii) `FLT` can flexibly discover generic client relatedness as well as an underlying clustered formation in non-IID scenarios; iii) we empirically show that `FLT` offers faster convergence compared to existing state-of-the-art baselines; iv) we provide convergence guarantees on `FLT` under common assumptions required for convergence of `FedAvg`; v) we demonstrate that `FLT` outperforms existing state-of-the-art baselines across several (synthetic and realistic) federated learning settings by about 3%, 9%, 2% and 40%, on MNIST, CIFAR10, FEMNIST, and a newly-introduced "Structured Non-IID FEMNIST" dataset); vi) finally, we show that `FLT` offers improved fairness (least variance in performance across clients), besides the improved accuracy.

2 Related Work

Federated learning of non-IID data. An early proposition to handle non-IID data in federated learning was to create a globally shared dataset comprised of a small subset of data from each client [23]. However, validity of this approach has been debated from increased communication costs, privacy, and security perspectives [3]. [13] proposed to add a proximal term to the local optimization sub-problems in order to limit the impact of "variable" local updates. This problem is phrased as "client drift" in [24] and a set of control variates are communicated between the clients and server to correct for this drift. In a somewhat related context but on different type of data (such as medical), other studies propose changing the global optimization cost-function or weighting the aggregation at the server side to account for dissimilarities among data distributions [25, 26].

Personalized and multi-task federated learning. This line of research tackles data heterogeneity by forming personalized models for clients via meta-learning or multi-task learning [15, 16, 14]. The work presented in [14] extends multi-task learning to federated learning; however, it relies on alternating bi-convex optimization which limits its applicability to only convex objective functions. In meta-learning setting [15, 16], first a single global model is obtained at the server, based on which each client fine-tunes its model. However, this global model would not serve as a good initialization if the client data distributions are considerably different.

Clustered federated learning. Akin to our high-level approach, two iterative approaches are proposed in [17, 18] in order to assign clients to separate clusters and train sub-models per cluster. However, several rounds of communication are required until the formation of clusters is solidified. More specifically, in [17] per iteration the cluster sub-models have to be sent to all the other active clients in that iterations, which is demanding in terms of communication cost. The clustering methodologies proposed in [19, 20, 27] are based on recursive bi-partitioning with cosine similarity between model updates as metric. Owing to the recursive nature, their compute and communication overhead can become a bottleneck in large-scale settings. A multi-center federated learning approach is proposed in [21] where clients are clustered based on their model parameter differences with randomly initialized cluster-level models. We argue that model parameters are just an implicit proxy of client data distributions, whereas our approach directly exploits the client data itself, and distills it into its signature encoding for clustering. Notably, the approach of [21] is sensitive to model initialization and also requires prior knowledge about the number of clusters. The cited approaches are mostly iterative and either slow in convergence or costly from communication perspective. We instead propose a one-shot solution based on client relatedness. In a concurrent work, [28] focuses on one-shot federated clustering. After projecting client data onto a selected subspace, the iterative Lloyd's k-means clustering [29] is applied and the outcomes is communicated to the server in a one-shot fashion for cluster assignment. Firstly, [28] does not study the impact of the proposed clustering on downstream federated learning applications. Besides, linear subspace decomposition is in practice less efficient than the non-linear (and fine-tuned) autoencoders we employ at client side. Moreover, our manifold learning approach with

Algorithm 1: Federated Averaging (`FedAvg`)

Require: $\overline{M}$, T, w^0, p_m
for $t = 0, \cdots, T-1$ **do**
  Server selects a subset $\mathcal{S}^t$ of clients ($\overline{M} = |\mathcal{S}^t|$);
  Server sends w^t to all the selected clients.
  for each client m **do**
    for epoch $e = 1, \cdots, E$ **do**
      $w_m \leftarrow w_m - \eta \nabla F_m(w_m)$
    end
    $w_m^{t+1} \leftarrow w_m$
  end
  Each client m sends w_m^{t+1} to the server.
  Server aggregates w_m's and updates w:
    $w^{t+1} \leftarrow \sum_{m=1}^{\overline{M}} p_m \, w_m^{t+1}$
end

`UMAP` [30] at the server side has the potential to approximate the underlying manifold of data in a more generic fashion as we empirically demonstrate in the following.

Notation. In the following, we use $\|X\|_l$ to denote norm-l of matrix X, and $|\mathcal{X}|$ to denote the cardinality of set $\mathcal{X}$. We denote the set $\{1, \cdots, n\}$ with $[n]$. We refer to the (i, j)-th element of matrix X as $X_{i,j}$ and its i-th row with X_i. $\mathbf{1}_n$ denotes a row vector of size n containing all 1's.

3 Federated Learning with Taskonomy

In this section, we first formalize the classical federated learning setting and objectives, and then introduce `FLT`.

3.1 Preliminaries: Refresher on `FedAvg`

In a classical federated learning setting, we consider M clients (in practice, hundreds) with n_m local data samples and communicating their learning regularly to a central server to reach global consensus about the whole data composed of $N = \sum_m n_m$ samples. In most prior work, the goal is to solve

$$\min_w f(w) = \sum_{m=1}^{M} p_m F_m(w), \tag{1}$$

where $p_m = \frac{n_m}{N}$ is the fraction of total data client m sees, and thus, $\sum_m p_m = 1$. The local objective F_m is typically defined by the empirical risk over local data $F_m = \frac{1}{n_m} \sum_{j=1}^{n_m} l_j(w)$. Federated learning is conducted in regular communication rounds (server to clients, and vice versa) and per round t typically a subset of clients $\mathcal{S}^t$ are randomly selected to run stochastic gradient descent (SGD) for a given number of local epochs. This *local updating* mechanism is shown to be more flexible and efficient than mini-batch methods [31, 32, 33]. Algorithm 1 summarizes `FedAvg` [4], a pioneering method to solve equation 1 in a non-convex setting. The protocol is simple: in T consecutive rounds, selected client m runs E epochs of SGD (with learning rate η) on local data and shares the local model w_m with the server to be averaged among $\overline{M}$ participating clients. For the sake of simplicity, the client participation fraction $\rho = \overline{M}/M = |\mathcal{S}^t|/|\mathcal{S}|$ is typically considered to be constant.

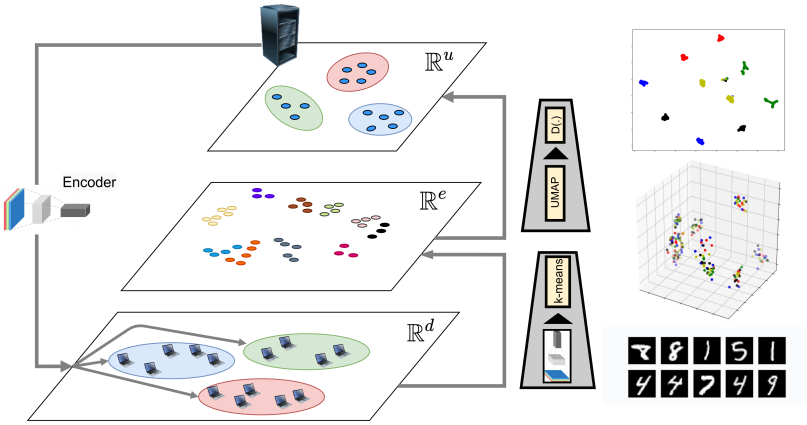


Figure 1: High-level architecture of FLT.

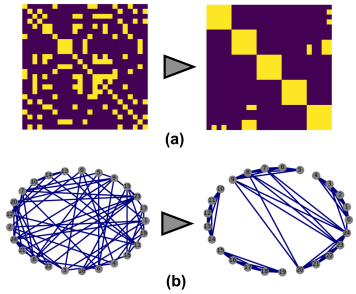


Figure 2: Forming $C = 5$ clusters for a network of $M = 25$ clients with hierarchical clustering. a) adjacency matrix and b) corresponding client relatedness graph (both reordered on the right).

3.2 Mechanics of FLT

Majority of the clustered federated learning approaches [17, 18, 19, 20, 27] enforce a hard membership constraint on the clients to form disjoint clusters where every client can belong to only one cluster. In contrast, we allow for an arbitrary symmetric task-relatedness matrix with the possibility to be reordered and relapsed into disjoint clusters. To form clusters, these approaches mostly compare the clients based on their model parameters using a distance metric (such as L_2 norm, or cosine similarity), which in practice does not capture the underlying manifold of data in the model parameter space or any other representation space. We instead propose a one-shot method for learning the task-relatedness matrix (coined as FLT) that benefits from manifold approximation (metric learning with UMAP [30]) at the server side before applying a distance metric. In the following, we delve deeper into the mechanics of FLT.

Overview. A high-level sketch of the proposed approach is depicted in Fig. 1. As can be seen, we consider three abstraction levels: i) data level, where data samples live in $\mathbb{R}^d$; ii) encoder level, where a contractive latent space representation of client data is extracted in an unsupervised fashion (samples are *nonlinearly* projected to $\mathbb{R}^e$); iii) manifold approximation level with UMAP, where samples live in $\mathbb{R}^u$. The encoder is provided by the server to the clients. This allows them to apply one-shot contractive encoding on their local data, followed by k-means on the outcome and return the results to the server. At server side, UMAP is applied to approximate the arriving clients embeddings. This is followed by applying a distance threshold to determine client dependencies and form an adjacency matrix or a client (task) relatedness graph. If forming disjoint clusters is of interest, we then use hierarchical clustering [34] to efficiently reorder the adjacency matrix (or corresponding client relatedness graph) into disjoint clusters (see Fig. 2).

Learning client (task) relatedness. The proposed approach, FCR, is described in Algorithm 2. The server broadcasts an encoder $G(\cdot)$ to all the M clients in $\mathcal{S}$. Note that this can also be an agreement (between the sever and clients) on using an encoder pretrained on a standard dataset (without virtually sending it), similar to the case of local models being used by clients in FedAvg. Nonetheless, this is *one-off*, and this downlink communication/agreement does not have to be repeated unless in an exceptional case where the server decides to change the architecture of the encoder. We considered a *simple* convolutional autoencoder (ConvAE) with its frozen encoder section employed for extracting latent embeddings. In our experimentation (Section 4), this model is as simple as the local client models for the downstream task. ConvAE not only helps compressing the information that has to be sent to the server, but also creates a less noisy representation of the client data. Upon receiving $G(\cdot)$ clients compute $\mathcal{E}_m := G(\mathcal{D}_m)$, where $\mathcal{D}_m$ denotes the dataset of client m ($\in [M]$) and $\mathcal{E}_m$ denotes its embedding set of size $|\mathcal{D}_m|$. The elements of $\mathcal{E}_m$ live in $\mathbb{R}^e$ with e referring to the latent embedding dimension. Even though $\mathcal{E}_m$ is compressed as compared to $\mathcal{D}_m$, it turns out that it can still be further distilled and yet capture enough information for our downstream federated learning purpose. Therefore, each client applies kMEANS$(\cdot)$ on $\mathcal{E}_m$ and sends the outcome $\mathcal{M}_m := \{\mu_1, \cdots, \mu_k\}$ (a set of size k) back to the server. The *fine-tune* mode is provisioned to accommodate encoders pretrained on a *totally different* dataset. In such a case, clients

will be asked to run F epochs of SGD from their latest state on their most recent dataset. As we will demonstrate in Section 4, this is to establish that the choice of dataset for pretraining the encoder is not a bottleneck.

The server then constructs $\mathcal{M} := \{\mathcal{M}_1, \cdots, \mathcal{M}_M\}$ and applies UMAP [30] to $\mathcal{M}$ and computes $\mathcal{Z} := \{\mathcal{Z}_1, \cdots, \mathcal{Z}_M\}$ with $\mathcal{Z}_m := \{u_{m,1}, \cdots, u_{m,k}\}$. $\mathcal{Z}$ contains $k \times M$ elements each living in $\mathbb{R}^u$, with u being typically 2 or 3. In most prior work, a distance metric (L_2 or cosine) is directly applied, which could be a limiting factor for non-convex risk functions and incongruent non-IID settings [19]. Instead, in FCR the server first learns the manifold in which the embeddings live using UMAP and then applies a distance metric to construct an adjacency matrix $A_{i,j} := \min_{r,s} \|u_{i,r} - u_{j,s}\|_2, \forall i, j \in [M]$ & $\forall r, s \in [k]$, where the minimum pairwise distance among the elements of $\mathcal{Z}_i$ and $\mathcal{Z}_j$ are taken into account. Here, for the sake of simplicity, we consider a *hard-thresholding* operator Γ applied on A leading to $\tilde{A}$, where $\tilde{A}_{i,j} = \Gamma(A_{i,j}) = \texttt{Sign}(A_{i,j} - \gamma)$ with γ a threshold value. If constructing *explicit and disjoint* clusters if of interest, the severs applies hierarchical clustering (HC) [34] to reorder $\tilde{A}$ into $\tilde{A}^r$ with C disjoint clusters (diagonal blocks). Finally, the server extracts cluster membership in $\mathcal{C} = \{\mathcal{C}_1, \cdots, \mathcal{C}_C\}$ with $\mathcal{C}_c$'s being a set of client ID's in cluster c. A flexibility that hierarchical clustering in [34] offers is to propose the best fitting number of clusters, if maximum number of clusters is not specified. $\tilde{A}$ and $\tilde{A}^r$ for a toy setup with $M = 25$ clients are depicted in Fig 2: a) adjacency matrix, b) client relatedness graph. On the left, the clients are arranged based on their ID's and on the right they are re-ordered with hierarchical clustering [34] to form $C = 5$ clusters. It is noteworthy that this one-shot client relatedness discovery of FCR can also happen in a few stages, in case all clients are not available for cooperation at the initialization stage. Besides, excluding a few clients from the process (for any reason) will not impact the performance of FLT..

Algorithm 2: Form Client Relatedness (FCR)

Require: MODE, $\mathcal{S}$, $G(\cdot)$, $\Gamma(\cdot)$, k, C
Server broadcasts $G(\cdot)$ to all clients in $\mathcal{S}$
for each client m in $\mathcal{S}$ **do**
  if MODE $=$ fine-tune **then**
    Client runs F epochs to fine-tune
  end
  Client computes its own embedding:
    $\mathcal{E}_m \leftarrow G(\mathcal{D}_m)$
  Client applies k-means clustering to $\mathcal{E}_m$'s:
    $\mathcal{M}_m := \{\mu_1, \cdots, \mu_k\} \leftarrow \texttt{kMEANS}(\mathcal{E}_m)$
  Client sends $\mathcal{M}_m$ to the server
end
Server updates $\mathcal{M} := \{\mathcal{M}_1, \cdots, \mathcal{M}_M\}$
Server (re)computes
    $\mathcal{Z} := \{\mathcal{Z}_1, \cdots, \mathcal{Z}_M\} \leftarrow \texttt{UMAP}(\mathcal{M})$
Server constructs the adjacency matrix:
  $A_{i,j} := \min_{r,s} \|u_{i,r} - u_{j,s}\|_2,$
    $\forall i, j \in [M]$ & $\forall r, s \in [k]$
Server applies thresholding $\tilde{A} := \Gamma(A)$
Server forms C clusters with hiearchical clustering:
    $\mathcal{C} = \{\mathcal{C}_1, \cdots, \mathcal{C}_C\} \leftarrow \texttt{HC}(\tilde{A})$
Return: $\tilde{A}$ and $\mathcal{C}$

Algorithm 3: FLT

Require: $\mathcal{S}$, M, T, W^0, p_m
Initialize Clustering:
    $\tilde{A}, \mathcal{C} \leftarrow \texttt{FCR}(\text{normal}, \mathcal{S}, G(\cdot), \Gamma(\cdot), k)$
for $t = 0, \cdots, T-1$ **do**
  $w_1^t, \cdots, w_m^t \leftarrow W^t$
  Server selects a subset $\mathcal{S}^t$ of clients
  Server sends w^t to all clients in $\mathcal{S}^t$
  for each client m *in* $\mathcal{S}^t$ **do**
    for epoch $e = 1, \cdots, E$ **do**
      $w_m \leftarrow w_m - \eta \nabla F_m(w_m)$
    end
    $\bar{w}_m^{t+1} \leftarrow w_m$
  end
  Each client sends$\bar{w}_m^{t+1}$ to the server
  Server collects $\overline{W}^{t+1} = [\bar{w}_1^{t+1}, \cdots, \bar{w}_M^{t+1}]$
  Server aggregates and updates $\overline{W}^{t+1}$:
  Case I) Full adjacency matrix $\tilde{A}$:
    $W^{t+1} \leftarrow \overline{W}^{t+1} \tilde{A}\, \texttt{diag}(p_m / \|\tilde{A}_m\|_0)$
  Case II) Disjoint clusters based on $\mathcal{C}$:
    $w_c^{t+1} \leftarrow \frac{1}{|\mathcal{C}_c|} \sum_{m \in \mathcal{C}_c} p_m \bar{w}_m^{t+1}, \forall c \in [C]$
    $W^{t+1} = [\mathbf{1}_{|\mathcal{C}_1|} w_1^{t+1}, \cdots, \mathbf{1}_{|\mathcal{C}_C|} w_C^{t+1}]$
end

Federated averaging with taskonomy. FLT in Algorithm 3 starts with an initialization stage by calling the *normal* mode of FCR. This returns the adjacency (or client relatedness) matrix $\tilde{A}$ together with an optional cluster membership set $\mathcal{C}$ for the case of disjoint clustering. Next, the typical T rounds of communication akin to FedAvg will be run. The server sends across a model corresponding to each client in the (randomly) selected (time-varying) subset $\mathcal{S}^t$ (line 6).

Similar to FedAvg, the client participation rate is defined as $\rho = |\mathcal{S}^t|/|\mathcal{S}|$. In case of disjoint clustering, all clients in cluster $\mathcal{C}_c$ will be given the same model w_c^t. Each client run F epoch on its local data and sends back the updated *local* model $\bar{w}_m^{t+1}$. The top bar notation is used to denote client-side models. The server collects all the local models in $\overline{W}^{t+1} = [\bar{w}_1^{t+1}, \cdots, \bar{w}_M^{t+1}]$ and updates them according to two possible cases. The first option (Case I) uses the full client relatedness matrix $\tilde{A}$ to benefit from the all the related client models (line 17). In this case, the sever updates the local model weights using $W^{t+1} = \overline{W}^t \, \tilde{A} \, \texttt{diag}(p_m / \|\tilde{A}_m\|_0)$. For notation simplicity, each model parameter set w_m^{t+1} is assumed to be reshaped into a row vector. The second option (case II) is to define an update rule per disjoint cluster according to cluster membership in $\mathcal{C}$. In that case, standard FedAvg will be applied per cluster (line 19) and the aggregated model of the cluster, w_c^{t+1}, will be sent to all the clients in the cluster in the next round (line 20).

Convergence of FLT. In Appendix D, we also provide a detailed convergence analysis for FLT under common assumptions required for the convergence of FedAvg.

4 Evaluation

We now present empirical evidence on the impact of the proposed approach, FLT. We first describe our experimental setup covering a suite of synthetic and realistic (large-scale) non-IID scenarios. We then compare the performance (accuracy and fairness) of FLT against existing state-of-the-art baselines.

4.1 Experimental Setup

Datasets. We opt for image classification task as our downstream application of federated learning. For performance evaluation, we employ four different datasets: i) MNIST [35], ii) CIFAR10 [36], iii) Federated EMNIST (FEMNIST) of LEAF [37] which is made out of Extended MNIST (EMNIST) [38], iv) and our newly designed *Structured Non-IID FEMNIST*. The latter is a resampled version of EMNIST to assess the performance in structured non-IID scenarios; see Appendix C.4 for more detailed description of this new dataset. For experiments on MNIST and CIFAR10, we respectively use $60,000$ and $50,000$ samples for training, and $10,000$ for testing. MNIST and CIFAR10 contain 10 classes of handwritten digits and objects with 6000 and 5000 samples per class, respectively. FEMNIST is a standard federated learning image classification dataset with $805,263$ samples that can accommodate up to 3550 clients. Based upon EMNIST and similar to FEMNIST, we build a new dataset with (more pronounced) structured non-IID conditions and call it Structured Non-IID FEMNIST. To do so, we consider the "balanced" dataset of EMNIST, containing $131,600$ samples on 47 classes. We use $112,800$ for training (2400 samples per class) and the remainder $18,800$ for testing. Depending on the scenarios explained later on, we partition the data into $C = 5$ or 10 clusters. For experiments on FEMNIST, there is no predefined partitioning (or clustering) and we follow the standard definitions of LEAF [37].

Encoders. To investigate the impact of encoder initialization, we also make use of three other datasets: a subset of CIFAR100 [39] (without any overlap with 10 classes of CIFAR10), a 20-class dataset composed of CIFAR10 and 10 classes of CIFAR100 (we refer to this as "CIFAR20"), and Fashion MNIST [40]. More concretely, we evaluate the performance of the proposed method in two encoder scenarios:

i) **Enc1:** the encoder provided to the clients is pretrained on a large set of targets that covers the client target classes. This case is hoping for a holistic encoder on the server side. More specifically, for federated learning on MNIST we have used an encoder pretrained on EMNIST, and for CIFAR10 we use CIFAR20 introduced earlier.

ii) **Enc2:** the encoder is pretrained on a totally different dataset, and an initial fine-tuning per client would be required. This case demonstrates that lacking the holistic encoder (Enc1) is not a bottleneck for FLT. Note that the encoder (ConvAE) is trained in an *unsupervised* way and independent of the downstream federated learning task. In this case, for federated learning on MNIST we use an encoder pretrained on Fashion MNIST, and for CIFAR10 we use CIFAR100, which has no overlap with CIFAR10. For experiments on EMNIST/FEMNIST and Structured Non-IID FEMNIST, we only considered Enc2 pretrained on Fashion MNIST.

Network parameters. We consider two model architectures for local client training, a multi-layer perceptron (MLP), and a convolutional neural network (CNN). We use an MLP with ReLU activation, and a single hidden layer of size 200. We use the CNN of LEAF [37] with 2 convolutional layers followed by 2 fully connected layers. The encoder section of our frozen ConvAE has 2 convolutional layers followed by a single fully-connected layer. Notice that the encoder is as simple as the local client model. See Appendix C.3 for more details on model architectures. We set the number of local epochs to $E = 5$, and the total communication rounds to $T = 100$, unless otherwise mentioned. The local training is a mini-batch SGD with batch size of 10 and learning rate $\eta = 0.01$. For `FLT`, the size of the latent embedding is $e = 128$ and k in `kMEANS` is set to 5. Even though on the client side k can be adjusted according to the number of client classes. The number of fine-tuning epochs for Enc2 is set to $F = 5$, and $\gamma = 1$. The client participation fraction $\rho = 20\%$, unless otherwise mentioned. For hierarchical clustering, we use the Ward's method for linkage creation [34].

4.2 Evaluation Scenarios

According to [3, 4], there are several possible sources of non-IIDness in client data distributions. Among those, label distribution skew, or so-called "pathological" partitioning is the most commonly adopted approach in literature. In this case, different clients will have different class labels, which together with quantity skew (varying number of samples across clients) leads to most destructive impact on `FedAvg` [41]. We build the following scenarios upon these angles.

Scenario 1 [MNIST, CIFAR10]: We consider a network of $M = |\mathcal{S}| = 100$ clients clustered into $C = 5$ clusters. The training data samples will be evenly distributed among these clients, 600 samples each (500 for CIFAR10), and the clients in each cluster will have samples only from two distinct classes. For instance, for MNIST, clients in cluster 1 ($\mathcal{C}_1$) will have only samples from digits "`0`" and "`1`", and those in $\mathcal{C}_2$ from "`4`" and "`7`" without overlap with $\mathcal{C}_1$.

Scenario2 [MNIST, CIFAR10]: This scenario is the same as scenario 1, except that the clients in two different clusters can have 1 similar label/class. As an example, for MNIST, clients in cluster 1 ($\mathcal{C}_1$) will have only samples from digits "`0`", "`1`", and "`2`", and those in $\mathcal{C}_2$ can have samples from "`2`", "`3`" and "`7`". This is to investigate the performance in less extreme non-IID conditions.

Scenario 3 [FEMNIST]: We import the standard FEMNIST dataset of LEAF and construct a network of 200 clients according to train and test data distributions defined in [37]. We run our experiments for a total of $T = 100$, 1000 and 1500 communication rounds. There are no predefined (or structured) clusters in FEMNIST and it is up to the federated learning method to form clusters, if need be. Here we consider both cases in Algorithm 3, and assess the performance of `FLT` in the generic form (no clusters, case I) as well as with $C = 2$, 3, 5 and 7 clusters (case II).

Scenario 4 [Structured Non-IID FEMNIST]: As mentioned earlier, we introduced this dataset with the purpose of imposing (more extreme) structured non-IIDness in FEMNIST. To this aim, we impose label distribution skew (across clusters) as well as quantity skew following a power law for the number of samples per client in each cluster, akin to [13]. We consider $C = 10$ clusters, each containing 5 distinct character classes (total of $12,000$ data samples per cluster), except the last one containing 2 classes (4800 samples), resulting in a total of 47 classes and $112,800$ samples. We also consider even a larger network than Scenario 3 with $M = 2400$ clients (240 clients per cluster) for MLP, and $M = 600$ clients (100 clients per cluster) for CNN. Notably, as a result of our random sampling strategy (Appendix C.4), the clients in each cluster will have a random subset of the labels assigned to that cluster.

4.3 Baselines and Competitors

Baselines. We consider 5 baselines as described in the following. i) `FedAvg` [4] where a *single global model* is trained for the whole network. ii) `local` where each client trains its own model with its own local data. iii) `PCA+kM+HC` Inspired by [28] (focused on federated clustering), this method applies linear principle component analysis (PCA) followed by `kMEANS` (at client-side) and hierarchical clustering (`HC`) at the server side. The goal is to illustrate

Table 1: Test accuracies (%± std. error) for Scenario 1.

	MNIST		CIFAR10	
Method	acc.	var.	acc.	var.
FedAvg	82.73±0.38	44.65	33.29±0.47	108.20
Local	97.41±0.16	3.44	79.81±0.40	115.35
PCA+kM+HC	83.82±0.37	41.13	75.35±0.40	589.6
FedSEM	98.01±0.14	2.12	78.23±0.41	695.01
FLT (Enc1)	97.98±0.14	2.17	87.11±0.33	88.80
FLT (Enc2)	97.97±0.14	2.17	87.17±0.33	84.62

Table 2: Test accuracies (%± std. error) for Scenario 2.

	MNIST		CIFAR10	
Method	acc.	var.	acc.	var.
FedAvg	86.08±0.35	17.55	52.77±0.50	19.18
Local	96.58±0.17	1.81	70.44±0.46	72.29
PCA+kM+HC	85.45±0.35	18.42	61.75±0.49	312.22
FedSEM	94.14±0.23	1.20	73.26±0.44	339.39
FLT (Enc1)	97.37±0.16	1.07	79.971±0.40	146.84
FLT (Enc2)	97.37±0.16	1.07	80.00±0.40	78.66

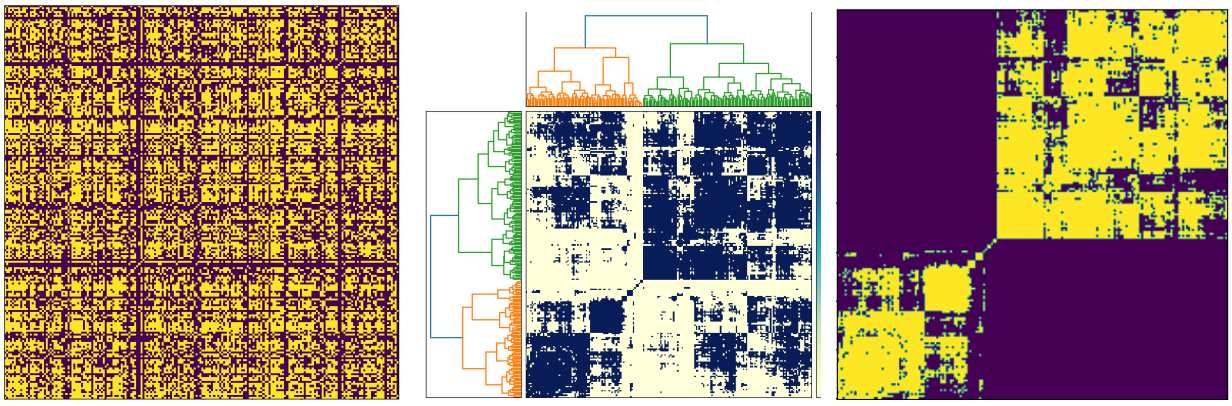
the impact of the nonlinear encoder (convAE) and manifold learning (UMAP) components of FLT for discovering task-relatedness. iv, v) We also compare our performance with two of the most recent state-of-the-art clustered federated learning approaches called IFCA [17] and FedSEM [21]. Notably, FedSEM already outperforms other recent baselines such as FedProx [13] and CFL [5], and thus, has been selected as the outstanding approach. FedSEM constructs multiple clusters each building its own local cluster-level model. For each cluster, a virtual cluster center is defined and its parameters are updated in an iterative fashion. The key idea behind cluster formation is measuring the distance between clients and the virtual cluster center model parameters. IFCA tries to construct multiple clusters iteratively by alternating between cluster identification estimation and loss function minimization. It starts with initializing model parameters for cluster centers which are then broadcast to the randomly participating clients per communication round (a costly communication overhead). The clients estimate their cluster identity by finding the model that returns the lowest loss, and send their cluster identity along with the updated model. For both methods: i) the number of clusters has to be known apriori; ii) a good initialization of cluster center model parameters is key for convergence; iii) they require several communication rounds before clusters are formed. We will demonstrate in the next subsections that these characteristics can lead to slow convergence and sub-optimal performance in *structured non-IID* scenarios.

Fairness in federated learning. Recently, *fairness* in performance has become an important concern in federated learning. In this context, being "fair" is to avoid disproportionately advantaging or disadvantaging some of the clients. Among several interesting approaches to fairness, two recent ones stand out in the federated learning literature: i) best worst-case performance [42, 8], and ii) least variance across clients [9]. We focus on the latter and report the variance of model accuracies across clients as a measure of fairness.

4.4 Evaluation Results for Scenarios 1 and 2

Average model accuracies on *test* data (and their standard errors) for Scenarios 1 and 2 after $T = 100$ communication rounds are shown in Tables 1 and 2. These are accompanied by the variance of model accuracies across clients which is serving as our fairness metric [9]. For MNIST, the number of local epochs is set to $E = 1$ and for CIFAR10 is set to $E = 5$. Note that irrespective of the choice of encoder (Enc1 or Enc2), the proposed method (FLT) achieves roughly the same performance in terms of accuracy in both scenarios. This highlights that our method is reasonably robust against the choice of the encoder and the fine-tuning idea (starting from a totally different dataset and finetuning only for $F = 5$ epochs) for Enc2 is functioning. As can be seen in both tables, even with $E = 1$, the differences between FLT and FedSEM are relatively smaller on MNIST but this discrepancy grows for CIFAR10. In summary, on MNIST and CIFAR10, we improve upon the state-of-the-art approach FedSEM by about 3% and 9% respectively, and way beyond that for the case of FedAvg, Local and PCA+kM+HC. On top of accuracy, FLT returns the least test accuracy variance (best fairness) for almost all settings in the two scenarios.

There are two reasons behind achieving reasonable performance by Local models: i) in these two scenarios clients have a reasonably high number of samples (600 for MNIST, 500 for CIFAR10); ii) the target classes in different clusters are almost independent. Notice that extreme non-IIDness (total class independence across clusters) in Scenario 1 helps Local models to score better than in Scenario 2 where some correlation between labels is allowed. A counter-argument applies to our method FLT because hard thresholding (mapping cluster membership to 0's and 1's) for Γ



(a) Raw adjacency matrix. (b) Reordered with hierarchical clustering. (c) Flattened to $C = 2$ clusters.

Figure 3: Hierarchical clustering (HC) is used to reorder clients relatedness graph ($\tilde{A} \rightarrow \tilde{A}^r$) and extract near-optimal block-diagonal structures for disjoint cluster formation. In (b), the dendrogram highlights the client dependencies leading to $C = 2$ disjoint clusters.

in Algorithm 2 seems to be limiting the performance in handling inter-cluster dependencies. Local models do not illustrate competitive performance. Therefore, we omit them in the following experiments.

4.5 Evaluation Results for Scenarios 3

Predefined data distribution of FEMNIST in LEAF [37] and its adoption by other state-of-the-art methods helped us to also benchmark our approach against IFCA [17]. Here, we present performance results based on both MLP and CNN networks for $M = 200$ clients; a setting commonly adopted in literature. For FLT, we only present the more challenging Enc2 case where the encoder is pretrained on Fashion MNIST, and in a *one-off process* each client fine-tunes the encoder (for only $F = 5$ epochs). The authors of [17] employ a slightly modified data sampling strategy; however, we use the standard data distribution of FEMNIST as in [37]. They also run for a total of $T = 6000$ communication rounds to reach the nominal performance, 2000 of which is initialization with FedAvg for "weight sharing" for a smaller client participation rate ($\rho = 3\%$). For the sake of a fair comparison, we run 500 rounds of initialization (with FedAvg) followed by another 1000 rounds of IFCA itself, i.e. a total of 1500 rounds. All the MLP experiments are run for $T = 1000$ communication rounds, and those for CNN are run for only $T = 100$ rounds, except for IFCA which is again run for 1500 rounds for both MLP and CNN. Remember that FEMNIST does not introduce any predefined cluster structure, and thus, non-IIDness in the structured form. As such, this is up to the methodology to define clusters. Both IFCA and FedSEM must define clusters ($C > 1$) or with $C = 1$ they would degenerate to FedAvg; however, thanks to FCR in Algorithm 2, this is not the case for FLT. We adopt the setting leading to the best performance reported by both IFCA and FedSEM with $C = 3$ and $C = 5$ clusters, respectively. For FLT we present the results for both cases described in Section 3: i) where the full client relatedness matrix will be used, and ii) where we use hierarchical clustering (HC) to specifically extract disjoint clusters (only if need be, as a special case). This is shown in Fig. 3 where (a) shows the raw client relatedness matrix, (b) illustrates how HC would reorder this to extract cluster level dependencies (dendrogram), and (c) shows the flattened version with $C = 2$ disjoint clusters.

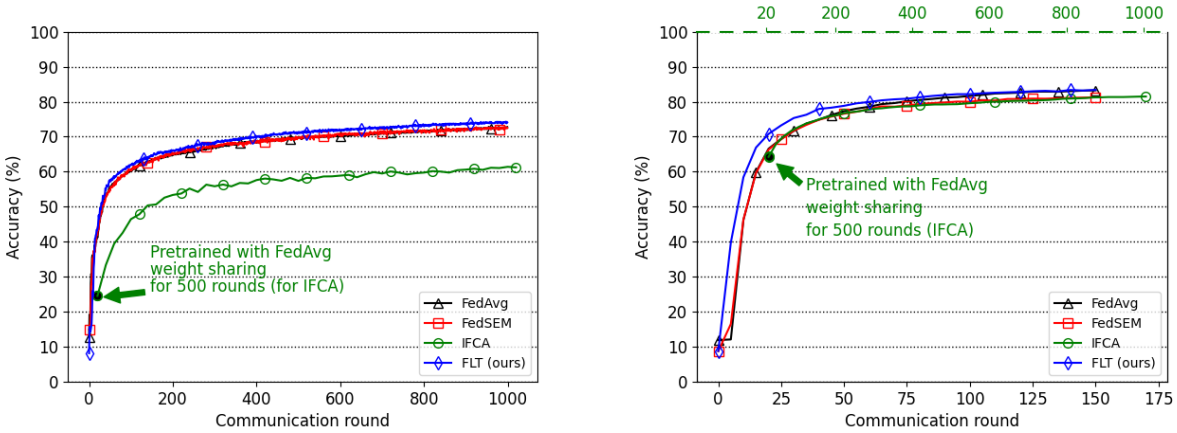
The *test* accuracies and fairness measures are summarized in Table 3. By using the full client relatedness matrix, FLT beats all the competitors (by $+2\%$, $+12\%$, $+1.5\%$ for PCA+kM+HC, IFCA, FedSEM) for MLP and (by $+2\%$, $+0.5\%$, $+2\%$ for PCA+kM+HC, IFCA, FedSEM) in case of CNN local models. Notably, we significantly outperform IFCA in the MLP setting and marginally outperform in the case of CNN. However, in case of CNN, all models except IFCA are run for only 100 communication rounds. In practice, it took IFCA 1500 communication rounds to reach a comparable performance regime. For this reason, we omit IFCA in our next experiments. Note that PCA+kM+HC (inspired by the line of thought in [28] and embedded in an end-to-end federated learning setting) also performs in par with the best

Table 3: Test accuracies (%± std. error) for Scenario 3.

Method	MLP		CNN	
	acc.	var.	acc.	var.
FedAvg	72.76±0.76	202.61	81.64±0.66	147.03
PCA+kM+HC($C=3$)	71.94±0.77	194.75	80.07±0.68	160.32
IFCA [17]	61.24±0.84	176.38	81.47±0.66	118.71
FedSEM [21]	72.45±0.76	185.96	79.99±0.68	156.27
FLT	74.11±0.74	171.31	82.14±0.65	145.03
FLT($C=2$)	72.83±0.76	175.55	80.53±0.68	150.26
FLT($C=3$)	72.61±0.76	184.59	79.72±0.69	170.10
FLT($C=5$)	72.05±0.77	180.32	78.71±0.70	169.22

Table 4: Test accuracies (%± std. error) for Scenario 4.

Method	MLP		CNN	
	acc.	var.	acc.	var.
FedAvg	46.50±0.36	100.27	53.58±0.36	883.47
PCA+kM+HC	48.75±0.36	467.92	59.87±0.36	912.39
FedSEM [21]	43.53±0.36	406.60	76.22±0.29	1264.38
FLT	86.51±0.24	207.60	93.69±0.10	98.65



(a) Test accuracies for MLP.

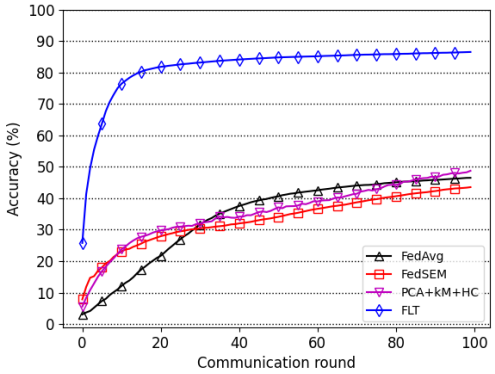
(b) Test accuracies for CNN.

Figure 4: Convergence graph of *test* accuracies for Scenario 3, FEMNIST, $M = 200$ (left: MLP, right, CNN). On the right for CNN, note the different range of communication rounds on the top horizontal axis associated with IFCA.

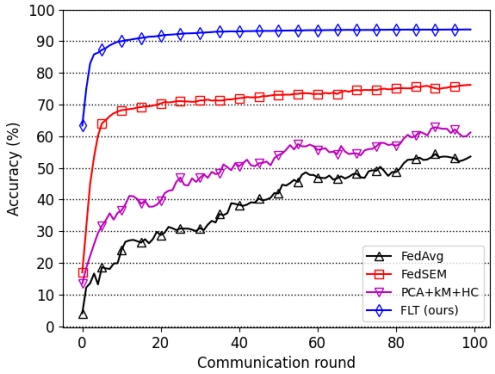
methods and considerably better than IFCA in the MLP setting. For the sake of fair comparison, hierarchical clustering (HC) with $C = 3$ clusters is applied here. The convergence graphs of average *test* accuracies are illustrated in Figs. 4a and 4b where FLT is the fastest in terms of convergence. We also investigated the impact of creating disjoint clusters with FLT (case II in Algorithm 3) in this setting. Introducing clusters ($C = 2$, 3 and 5) in this dataset seems to be slightly degrading the performance of FLT, even though it still remains to be in par with the best performing models. See Appendix A for more detailed results. Reflecting on the relatively smaller performance margin in this scenario, we argue that FEMNIST may not have a clear cluster *structure*, and thus, cluster-based methods might not offer a significant gain. This also concurs with that FLT employing the full adjacency matrix slightly outperforms its clustered variants. This was the main motivation behind designing Scenario 4 to assess the potential of these algorithms in a more challenging large-scale setting: Structured Non-IID FEMNIST.

4.6 Evaluation Results for Scenarios 4

As explained earlier, Scenario 4 presents a large-scale federated learning setting with structured non-IIDness involving both quantity and label distribution skews. For FLT, we only present the more challenging Enc2 case where the encoder is pretrained on Fashion MNIST, and in a *one-off process* each client fine-tunes the encoder (for only $F = 5$ epochs). Convergence graphs of average *test* accuracies are shown in Fig. 5a and 5b, for MLP and CNN networks, respectively. The *test* accuracies (and their standard error) together with the fairness measure (variance across clients) for at the last communication round $T = 100$ are summarized in Table 4. Interestingly, the state-of-the-art competitor,



(a) Test accuracies for MLP, $M = 2400$.



(b) Test accuracies for CNN, $M = 600$.

Figure 5: Convergence graph of *test* accuracies for Scenario 4, Structured Non-IID FEMNIST, $C = 10$.

FedSEM, suffers in the MLP scenario and roughly scores as good as the barebone FedAvg, which is not made to cope with structured non-IID scenarios. The performance of FedSEM is noticeably improved in the CNN setting, but accordingly FLT is boosted as well, reaching $+93\%$ *test* accuracy in this challenging scenario. Here, we beat FedSEM by $+40\%$ and $+17\%$ in MLP and CNN scenarios, respectively. The main reason behind this downgrade in performance of FedSEM (compared to Scenario 3) is the struggle to discover and form clusters by relying on model parameter comparisons. This is in turn due to two main factors: i) significantly larger number of models (per cluster and across) for this extreme non-IID scenario leads to tremendous heterogeneity in model space and thus considerable increase in complexity of pairwise model comparisons; ii) the problem is exacerbated due to the limited number of samples provided to some clients (down to 20 samples because of the power law) resulting in lower-quality local model training. An evidence confirming this hypothesis is that these problems are even more pronounced in the case of a simpler local model (MLP), where FedSEM falls almost back to FedAvg. On the other hand, thanks to FCR on client side, and UMAP and HC) on server side, FLT manages to automatically detect $C = 10$ clusters (See Appendix B for more results). Both FedAvg and FedSEM can benefit from more communication rounds in this scenario. Nonetheless, the main message of this experiment is clear: the one-shot taskonomy-based client relatedness helps FLT to converge already in roughly 20 communication rounds and outperform the other baselines by a significant margin, while also offering a lower variance across clients (best fairness) compared to most competitors.

4.7 Impact of Hyper-parameter Changes

We investigate the impact of a few remaining important hyper-parameters on the performance of different FedSEM and FLT. We ran this experiment on Scenario 2 due to being an intermediate case where label overlap is allowed. Following this scenario, we set $C = 5$, and focus on CIFAR10, the more challenging dataset. The results are summarized in Table 5. Here, we consider $M = 100$ and 500, client participation fraction per communication round $\rho = 20\%$ and 50% (denoted as 0.2 and 0.5), $E = 5$, and we report the *test* accuracy after $T = 50$ and 100 communication rounds. As can be seen in this table, by keeping the total number of clusters C constant, increasing the total number of clients M and the participation fraction ρ improves the overall performance of both methods. Without exception, our method outperforms FedSEM by about 5% to 9%.

Table 5: Ablation study of *test* accuracies (%) for Scenario 2

(M, ρ, E, T)	FedSEM [21]	FLT
(100, 0.2, 5, 50)	71.85±0.45	79.45±0.40
(100, 0.2, 5, 100)	73.42±0.44	79.95±0.40
(100, 0.5, 5, 50)	74.29±0.44	80.02±0.40
(100, 0.5, 5, 100)	74.15±0.44	79.72±0.40
(500, 0.2, 5, 50)	71.24±0.44	80.27±0.40
(500, 0.2, 5, 100)	73.21±0.45	81.77±0.39
(500, 0.5, 5, 50)	76.79±0.42	81.97±0.37
(500, 0.5, 5, 100)	76.75±0.42	81.57±0.39

Table 6: Communication complexity analysis.

FLT	FedSEM [21]	IFCA [17]
$\underbrace{M * W_{\text{enc}} + k\,M\,e}_{\text{one-off}} + 2\rho M W_{\text{local}} T$	$2\rho M W_{\text{local}} T$	$\rho M W_{\text{local}} T(C+1)$

5 Concluding Remarks

Summary and convergence analysis. We proposed FLT that comes with the following notable advantages. First, it is one-shot and considerably faster in convergence compared to its competitors, especially in structured non-IID scenarios. Second, the problem formulation of FLT can handle both generic client relatedness (no specific clustering) as well as disjoint clusters, if need be. Third, in contrast to most existing baselines, it does not require prior knowledge about number of clusters to form them. Fourth, it performs slightly better than the state-of-the-art baselines in standard federated learning settings and significantly outperforms them in structured non-IID scenarios. Finally, FLT offers improved fairness (least performance disparity among clients) compared to the existing baselines in most presented scenarios. Last but not least, in Appendix D, we also provide a detailed convergence proof for FLT under common assumptions required for the convergence of FedAvg.

Complexity and practical considerations. FLT introduces a *one-off* overhead due to the client relatedness discovery process (FCR, Algorithm 2). However, owing to FCR, it is faster than the existing iterative baselines and less prone to convergence issues, as we have demonstrated throughout Section 4 and Appendix B. One can argue that this step can be prone to security issues during the uplink communication (akin to standard FedAvg communications). A possible solution to address this is adding encryption and client ID verification processes, which are outside the scope of our work. From communication complexity perspective, this overhead requires the server to send an encoder model (W_{enc}) to the clients, and the clients to send an array of size ke (with k in k-means and e denoting the latent embedding dimension of the encoder) to the server. A rough estimate of the communication complexity of the proposed FLT, and two discussed state-of-the-art competitors (FedSEM and IFCA) is summarized Table 6. As can be seen, the communication complexity of FLT and FedSEM are essentially the same except for the first two one-off terms (without T for total communication rounds) which could be neglected. Note that this is an initialization step and it can also happen in multiple steps. Excluding a few clients from this process due to for instance their unavailability, does not impact the performance of FCR and in turn FLT. On the other hand, IFCA mandates roughly $(C+1)/2$ times (C being the number of clusters) more communication complexity. This is because in every communication round, C virtual center models will have to be sent to all the participating clients. From compute complexity perspective, possible *fine-tuning* of the encoder is only for a small number of epochs ($F = 5$ in our experiments) on an encoder which is as simple as the local client models; and this is yet another one-off process that can be neglected over long runs. When the number of clients grows to tens of thousands (in very large-scale networks), FLT has the flexibility to decompose the client relatedness graph into disjoint clusters and degenerate to the same complexity level its competitors inflict.

6 Acknowledgment

The authors would like to thank Shell Global Solutions International B.V. and Delft University of Technology (TU Delft) for their support and for the permission to publish this work. The authors also thank Ahmad Beirami from Facebook AI for helpful discussions.

References

[1] J. Konečnỳ, B. McMahan, and D. Ramage, "Federated optimization: Distributed optimization beyond the datacenter," *arXiv preprint arXiv:1511.03575*, 2015.

[2] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[3] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, *et al.*, "Advances and open problems in federated learning," *arXiv preprint arXiv:1912.04977*, 2019.

[4] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*, pp. 1273–1282, PMLR, 2017.

[5] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *IEEE transactions on neural networks and learning systems*, 2019.

[6] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *arXiv preprint arXiv:1712.07557*, 2017.

[7] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *International Conference on Artificial Intelligence and Statistics*, pp. 2938–2948, PMLR, 2020.

[8] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," *arXiv preprint arXiv:1902.00146*, 2019.

[9] T. Li, M. Sanjabi, A. Beirami, and V. Smith, "Fair resource allocation in federated learning," *arXiv preprint arXiv:1905.10497*, 2019.

[10] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, "Can you really backdoor federated learning?," *arXiv preprint arXiv:1911.07963*, 2019.

[11] H. Wang, K. Sreenivasan, S. Rajput, H. Vishwakarma, S. Agarwal, J.-y. Sohn, K. Lee, and D. Papailiopoulos, "Attack of the tails: Yes, you really can backdoor federated learning," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[12] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[13] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *arXiv preprint arXiv:1812.06127*, 2018.

[14] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," *Advances in neural information processing systems*, vol. 30, pp. 4424–4434, 2017.

[15] Y. Jiang, J. Konečnỳ, K. Rush, and S. Kannan, "Improving federated learning personalization via model agnostic meta learning," *arXiv preprint arXiv:1909.12488*, 2019.

[16] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning: A meta-learning approach," *arXiv preprint arXiv:2002.07948*, 2020.

[17] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," *arXiv preprint arXiv:2006.04088*, 2020.

[18] Y. Mansour, M. Mohri, J. Ro, and A. T. Suresh, "Three approaches for personalization with applications to federated learning," *arXiv preprint arXiv:2002.10619*, 2020.

[19] F. Sattler, K.-R. Müller, and W. Samek, "Clustered federated learning: model-agnostic distributed multitask optimization under privacy constraints," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[20] C. Briggs, Z. Fan, and P. Andras, "Federated learning with hierarchical clustering of local updates to improve training on non-iid data," *arXiv preprint arXiv:2004.11791*, 2020.

[21] M. Xie, G. Long, T. Shen, T. Zhou, X. Wang, and J. Jiang, "Multi-center federated learning," *arXiv preprint arXiv:2005.01026*, 2020.

[22] A. R. Zamir, A. Sax, W. Shen, L. J. Guibas, J. Malik, and S. Savarese, "Taskonomy: Disentangling task transfer learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3712–3722, 2018.

[23] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[24] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," in *International Conference on Machine Learning*, pp. 5132–5143, PMLR, 2020.

[25] Y. Yeganeh, A. Farshad, N. Navab, and S. Albarqouni, "Inverse distance aggregation for federated learning with non-iid data," in *Domain Adaptation and Representation Transfer, and Distributed and Collaborative Learning*, pp. 150–159, Springer, 2020.

[26] Y. Laguel, K. Pillutla, J. Malick, and Z. Harchaoui, "Device heterogeneity in federated learning: A superquantile approach," *arXiv preprint arXiv:2002.11223*, 2020.

[27] M. Duan, D. Liu, X. Ji, R. Liu, L. Liang, X. Chen, and Y. Tan, "Fedgroup: Ternary cosine similarity-based clustered federated learning framework toward high accuracy in heterogeneity data," *arXiv preprint arXiv:2010.06870*, 2020.

[28] D. K. Dennis and V. Smith, "Heterogeneity for the Win: Communication-Efficient Federated Clustering." `http://icfl.cc/wp-content/uploads/2020/12/SpicyFL_2020_paper_35.pdf`, 2020.

[29] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[30] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.

[31] S. U. Stich, "Local sgd converges fast and communicates little," *arXiv preprint arXiv:1805.09767*, 2018.

[32] J. Wang and G. Joshi, "Cooperative sgd: A unified framework for the design and analysis of communication-efficient sgd algorithms," *arXiv preprint arXiv:1808.07576*, 2018.

[33] B. E. Woodworth, J. Wang, A. Smith, B. McMahan, and N. Srebro, "Graph oracle models, lower bounds, and gaps for parallel stochastic optimization," *Advances in neural information processing systems*, vol. 31, pp. 8496–8506, 2018.

[34] D. Müllner, "Modern hierarchical, agglomerative clustering algorithms," *arXiv preprint arXiv:1109.2378*, 2011.

[35] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.

[36] A. Krizhevsky, V. Nair, and G. Hinton, "The cifar-10 dataset," *online: http://www. cs. toronto. edu/kriz/cifar. html*, vol. 55, 2014.

[37] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar, "Leaf: A benchmark for federated settings," *arXiv preprint arXiv:1812.01097*, 2018.

[38] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, "Emnist: Extending mnist to handwritten letters," in *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 2921–2926, IEEE, 2017.

[39] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," *Citeseer*, 2009.

[40] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[41] T.-M. H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," *arXiv preprint arXiv:1909.06335*, 2019.

[42] T. Hashimoto, M. Srivastava, H. Namkoong, and P. Liang, "Fairness without demographics in repeated loss minimization," in *International Conference on Machine Learning*, pp. 1929–1938, 2018.

[43] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *Journal of the American Statistical Association*, vol. 58, no. 301, pp. 13–30, 1963.

[44] S. Wang and T.-H. Chang, "Federated clustering via matrix factorization models: From model averaging to gradient sharing," 02 2020.

[45] J. Bolte, S. Sabach, and M. Teboulle, "Proximal alternating linearized minimization for nonconvex and nonsmooth problems," *Mathematical Programming*, vol. 146, 08 2013.

Table 7: Train and test accuracies ($\%\pm$ std. error) for Scenario 1.

	MNIST				CIFAR10			
	train		test		train		test	
Method	acc.	var.	acc.	var.	acc.	var.	acc.	var.
FedAvg	82.50±0.16	43.19	82.73±0.38	44.65	35.08±0.21	152.54	33.29±0.47	108.20
Local	99.25±0.04	0.60	97.41±0.16	3.44	99.87 ±0.02	0.80	79.81±0.40	115.35
PCA+kM+HC	83.65±0.16	49.2	83.82±0.37	41.13	78.07±0.19	589.65	75.35±0.40	589.6
FedSEM [21]	97.94±0.06	2.58	98.01±0.14	2.12	83.43 ±0.17	725.99	78.23±0.41	695.01
FLT (Enc1)	97.96±0.06	2.50	97.98±0.14	2.17	93.52±0.11	36.21	87.11±0.33	88.80
FLT (Enc2)	97.95±0.06	2.50	97.97±0.14	2.17	93.51 ±0.11	39.42	87.17±0.33	84.62

Table 8: Train and test accuracies ($\%\pm$ std. error) for Scenario 2.

	MNIST				CIFAR10			
	train		test		train		test	
Method	acc.	var.	acc.	var.	acc.	var.	acc.	var.
FedAvg	85.62±0.14	20.57	86.08±0.35	17.55	55.87±0.22	14.18	52.77±0.50	19.18
Local	99.17±0.04	0.36	96.58±0.17	1.81	100.0±0.0	0.0	70.44±0.46	72.29
PCA+kM+HC	85.27±0.15	19.09	85.45±0.35	18.40	68.75±0.21	398.82	61.75±0.49	312.22
FedSEM [21]	94.26±0.10	1.27	94.14±0.23	1.20	84.30±0.16	426.27	73.26±0.44	339.39
FLT (Enc1)	97.54±0.06	0.93	97.37±0.16	1.07	95.01±0.10	124.07	79.97±0.40	146.84
FLT (Enc2)	97.52±0.06	0.93	97.37±0.16	1.07	94.87±0.10	24.07	80.00±0.40	78.66

Table 9: Train and test accuracies ($\%\pm$ std. error) for Scenario 3.

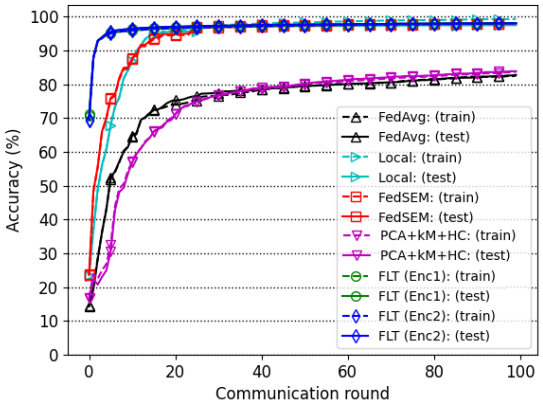
	MLP ($T = 1000$)				CNN ($T = 100$ except for IFCA with $T = 1500$)			
	train		test		train		test	
Method	acc.	var.	acc.	var.	acc.	var.	acc.	var.
FedAvg	74.43±0.25	100	72.76±0.76	202.61	82.96±0.22	81.28	81.64±0.66	147.03
PCA+kM+HC($C=3$)	75.77±0.25	111.44	71.94±0.77	194.75	83.99±0.21	89.92	80.07±0.68	160.32
IFCA [17]	61.57±0.28	48.70	61.24±0.84	176.38	84.03±0.21	35.64	81.47±0.66	118.71
FedSEM[21]	77.03±0.24	100.18	72.45±0.76	185.96	86.57±0.20	56.83	79.99±0.68	156.27
FLT	76.43±0.25	81.13	74.11±0.74	171.31	84.93±0.21	56.22	82.14±0.65	15.03
FLT($C=2$)	75.23±0.22	96.79	72.83±0.76	175.55	83.15±0.22	78.27	80.53±0.68	150.26
FLT($C=3$)	75.73±0.25	101.14	72.61±0.76	184.59	83.88±0.21	74.82	79.72±0.69	170.10
FLT($C=5$)	77.24±0.24	82.09	72.05±0.77	180.32	85.49±0.20	63.87	78.71±0.70	169.22

Table 10: Train and test accuracies ($\%\pm$ std. error) for Scenario 4.

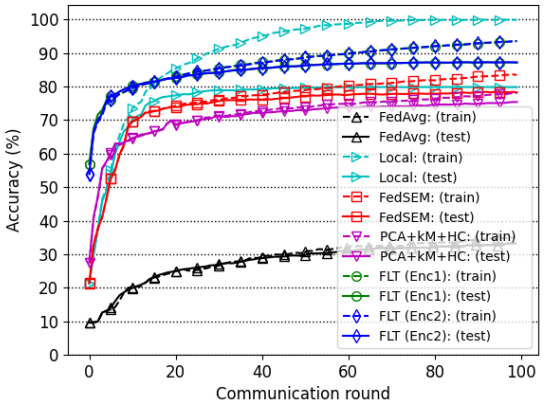
	MLP				CNN			
	train		test		train		test	
Method	acc.	var.	acc.	var.	acc.	var.	acc.	var.
FedAvg	45.48±0.15	281.95	46.50±0.36	100.27	56.19±0.15	908.67	53.58±0.36	883.47
PCA+kM+HC	72.66±0.13	770.84	48.75±0.36	467.92	62.92±0.14	863.94	59.87±0.36	912.39
FedSEM [21]	45.19±0.15	508.13	43.53±0.36	406.60	79.45±0.12	1295.72	76.22±0.29	1264.38
FLT	91.23±0.08	155.36	86.51±0.24	207.60	97.95±0.04	6.59	93.69±0.10	98.65

A More Detailed Evaluation Results

Tables 7, 8, 9, and 10 respectively summarize the statistics of our experimentation on Scenarios 1, 2, 3 and 4, described in Section 4. Here, we also report the *train* statistics for the sake of completeness. Note that the train accuracies can be extremely high, due to overfitting to the small number of samples clients can possibly have in some scenarios. To further clarify, in Table 9 (extended version of Table 3 on FEMNIST dataset) for the case of MLP, all the methods are run for $T = 1000$ communication rounds because the learning process is slower compared to the case of CNN. As can be seen, after $T = 1000$ communication rounds, the network hardly reaches the same level of performance it achieves

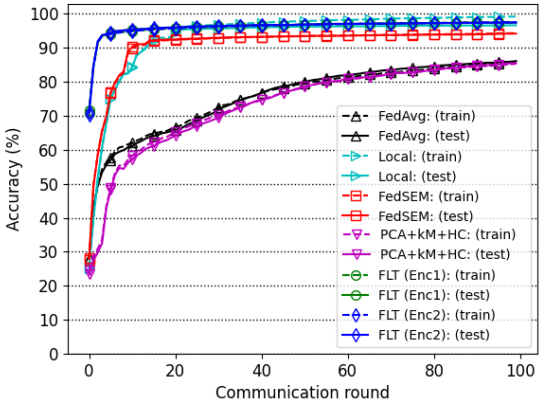


(a) Train and test accuracies for MNIST.

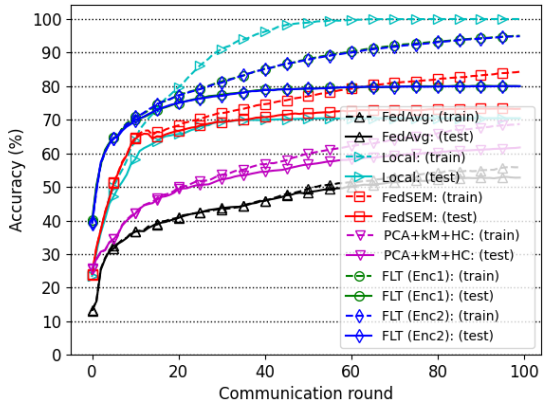


(b) Train and test accuracies for CIFAR10.

Figure 6: Train and test accuracies for Scenario 1, $C = 5$, $M = 100$.



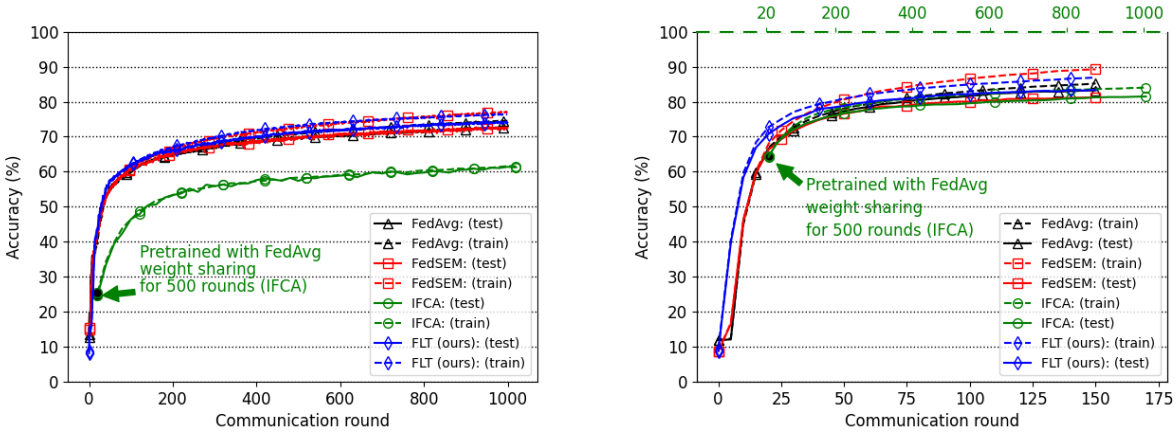
(a) Train and test accuracies for MNIST.



(b) Train and test accuracies for CIFAR10.

Figure 7: Train and test accuracies for Scenario 2, $C = 5$, $M = 100$.

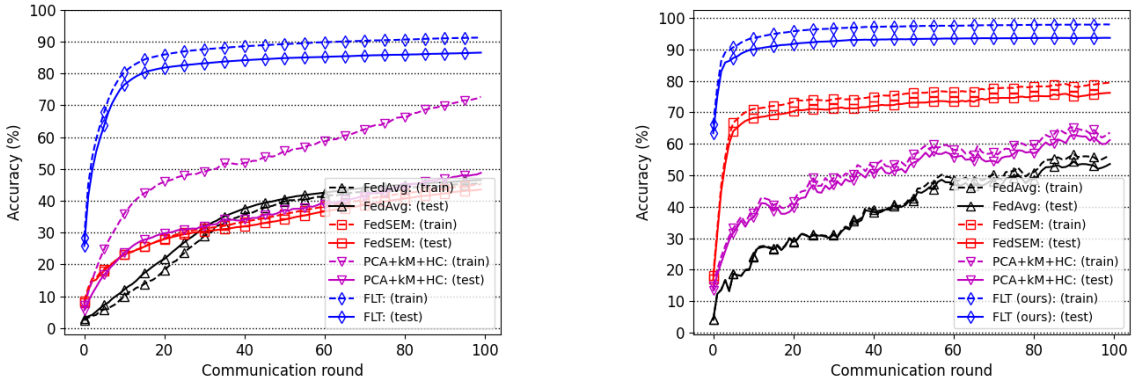
with only $T = 100$ rounds with CNN. The only exception here is `IFCA` [17] which seems to be slower in convergence in this setting irrespective of the choice of local models. To make it reach a comparable performance for CNN case, we exceptionally allowed `IFCA` to run $T = 1500$ rounds (15 times more) in contrast to $T = 100$ for the rest of the methods. This demonstrates the difference between the proposed approach (`FLT`) and `IFCA` in term of convergence speed. In this setting, `FLT` is at least 10 times faster than `IFCA`. The converge graphs corresponding to Tables 7, 8, 9, and 10 are respectively illustrated in Figs. 6, 7, 8 and 9. The *train* convergence graphs are plotted with dashed lines and with the same markers as the corresponding *test* curves. Note the different range of communication rounds on the top horizontal axis of Fig. 8 (for CNN on the right) associated with `IFCA`. On the results for Scenario 3, Fig. 10 provides a more complete picture of the impact of imposing a predefined number of clusters in hierarchical clustering (`HC`) on the performance of `FLT`. As can be seen, the test accuracies slightly degrade compared to `FLT` using the full client relatedness matrix. Increasing the number of clusters seems to have a gradual downgrading impact.



(a) Train and test accuracies for MLP.

(b) Train and test accuracies for CNN.

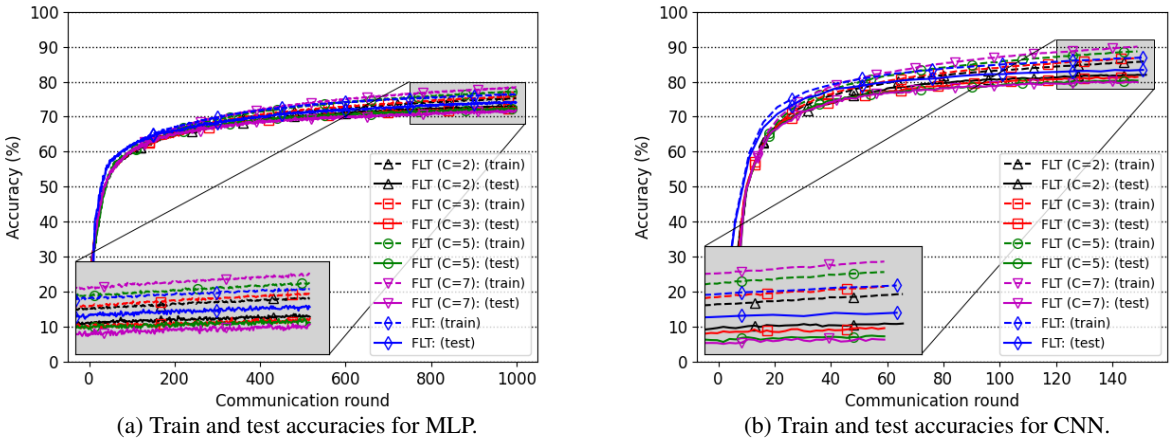
Figure 8: Convergence graph of *train* and *test* accuracies for Scenario 3, FEMNIST, $M = 200$. On the right for CNN, note the different range of communication rounds on the top horizontal axis associated with `IFCA`.



(a) Train and test accuracies for MLP, $M = 2400$.

(b) Train and test accuracies for CNN, $M = 600$.

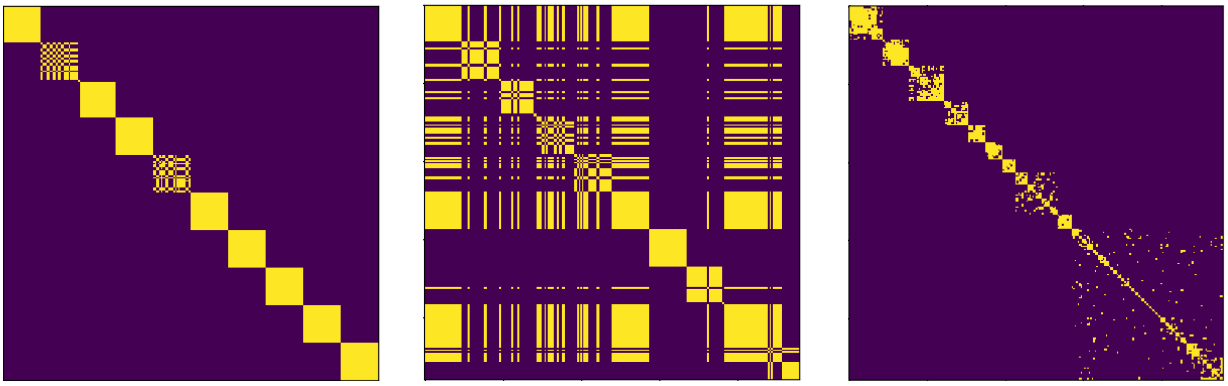
Figure 9: Convergence graph of *train* and *test* accuracies for Scenario 4, Structured Non-IID FEMNIST, $C = 10$.



(a) Train and test accuracies for MLP.

(b) Train and test accuracies for CNN.

Figure 10: Convergence graph of *train* and *test* accuracies for Scenario 3, FEMNIST, $M = 200$. Comparing the impact of imposing different number of clusters C for hierarchical clustering (`HC`) on the performance of `FLT`.



(a) FLT (b) FedSEM (c) PCA+kM+HC

Figure 11: One-shot cluster formation (re-ordered adjacency matrix corresponding to client relatedness graph) for FLT in Scenario 2 with $C = 10$ clusters and a total of $M = 240$ clients. The *structure* of the $C = 10$ clusters is almost perfectly estimated by Algorithm 2.

B Convergence Issues of Existing Iterative Baselines

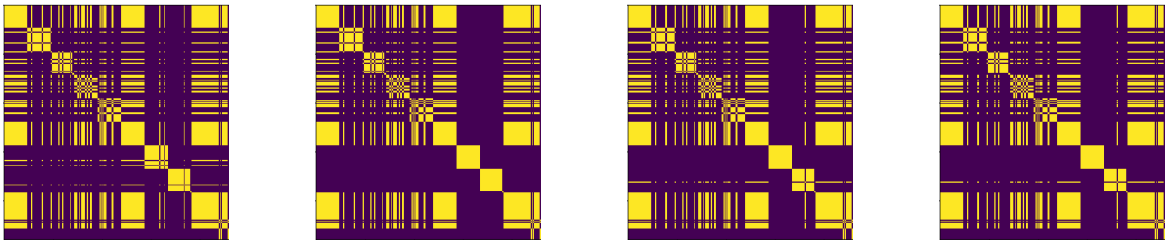
In Subsection 4.6, we compared the performance of the proposed method FLT against FedSEM [21] and PCA+kM+HC, which turned out to be the closest competitors according to the evaluation results of Scenarios 3. In Scenario 4, we proposed to challenge these methods on our newly introduced *Structured Non-IID FEMNIST* dataset with $C = 10$ predefined clusters each having different set of labels. Basically, in Scenario 4, we impose two levels of non-IIDness, quantity skew and structured label distribution skew. The results in Subsection 4.6 confirmed that both FedSEM and PCA+kM+HC cannot offer a competitive performance against FLT. For FedSEM and any other iterative model comparison based approach, we argued that this is due to tremendous heterogeneity in model space, and thus, considerable increase in complexity of pairwise model comparisons in such large-scale non-IID scenarios. When it comes to PCA+kM+HC, even though it is a one-shot approach (following the line of thought proposed in [28]), the client-side linear dimensionality reduction seems to be the bottleneck, even though it still benefits from hierarchical clustering and misses out on manifold learning with UMAP on the sever side. To further elaborate on this, in a relatively smaller setup with $M = 240$ clients, Fig. 11 demonstrates the efficiency of the proposed client relatedness formation process (FCR, Algorithm 2) in automatically discovering the cluster structure in Scenario 4. The figure (a) shows how FLT manages to almost perfectly group the clients into 10 clusters in the form of a block-diagonal adjacency matrix. On the other hand in (b) and (c), the cluster formation of FedSEM and PCA+kM+HC is far from ideal. The iterative process of cluster formation for FedSEM is visualized in Fig. 12. As can be seen, the cluster formation process does not improve beyond the communication round 15 and even gets slightly worse from there on. This is the reason behind the sub-optimal performance of FedSEM [21] in large-scale structured non-IID condition imposed in Scenario 4.

C Implementation Details

C.1 Machines

We simulated the federated learning settings on standard virtual machines (VMs) on Microsoft Azure cloud. Our server VM (NC6v3) includes 2 Intel(R) Xeon(R) E5-2690 CPUs (each has 6 vCPUs) and 2 NVidia V100 Tesla GPUs.

(a) Comm. round 1 (b) Comm. round 3 (c) Comm. round 6 (d) Comm. round 9



(e) Comm. round 12 (f) Comm. round 15 (g) Comm. round 18 (h) Comm. round 50

Figure 12: Iterative cluster formation discovery process of `FedSEM` in Scenario 2. The expected outcome is a block-diagonal matrix akin to the one presented in Fig. 11; however, `FedSEM` gets stuck at comm. round 15.

C.2 Software

The full code and all the associated experiments are publicly available at: `https://github.com/hjraad/FLT/`.

C.3 Model architectures

For reader's reference, below is detailed model architectures for the multi-layer perceptron (MLP) employed in local model training.

```
==========================================================================
Layer (type:depth-idx)    Specs
==========================================================================
Linear: 1-1               in_features=flatten(input), out_features=200
Dropout: 1-2              p=0.5
Linear: 1-3               in_features=200, out_features=num_classes
==========================================================================
Total params: 169,462
Trainable params: 169,462
Non-trainable params: 0
==========================================================================
```

We have used the CNN architecture proposed in the LEAF package [37], as summarized below.

```
==========================================================================
Layer (type:depth-idx)    Specs
==========================================================================
Conv2d: 1-1               channels=32, kernel_size=5, stride=1, padding=2
MaxPool2d: 1-2            kernel_size=2, stride=2, padding=0
Conv2d: 1-3               channels=63, kernel_size=5, stride=1, padding=2
Linear: 1-4               in_features=flatten(previous), out_features=2048
Linear: 1-5               in_features=2048, out_features=num_classes
==========================================================================
Total params: 3,459,582
Trainable params: 3,459,582
Non-trainable params: 0
==========================================================================
```

Algorithm 4: Structured Non-IID FEMNIST Sampler

Require: min number of samples per client α, power exponent of the power law δ, and cluster formation with $\mathcal{L} = \{\mathcal{L}_1, \cdots, \mathcal{L}_C\}, \mathcal{C} = \{\mathcal{C}_1, \cdots, \mathcal{C}_C\}$.

Sort the training data according to the labels assigned to clusters $\mathcal{D}^t = \{\mathcal{D}^t_{\mathcal{L}_1}, \cdots, \mathcal{D}^t_{\mathcal{L}_C}\}$

for each cluster $c \in [C]$ **do**

  $\beta^*_c = \arg\min_\beta(\sum_m \alpha + e^{\beta\, m^\delta} - |\mathcal{D}^t_{\mathcal{L}_c}|); \forall m \in [M_c]$, with $M_c := |\mathcal{C}_c|$

  Randomly shuffle samples in $\mathcal{D}^t_{\mathcal{L}_c}$

  for each client $m \in [M_c]$ **do**

    Sequentially assign a batch of $\alpha + e^{\beta^*_c\, m^\delta}$ data samples from $\mathcal{D}^t_{\mathcal{L}_c}$

  end

end

Finally, below we have our main convolutional Autoencoder (ConvAE) which is either trained apriori, or will be fine-tuned on client data. After training or fine-tuning, the encoder section of the ConvAE is used for extracting embeddings which will in turn be used for the formation of client relatedness graph in Algorithm 2.

```
==========================================================================
Layer (type:depth-idx)    Specs
==========================================================================
Conv2d: 1-1               channels=16, kernel_size=3, stride=1, padding=1
Conv2d: 1-2               channels=4, kernel_size=3, stride=1, padding=1
MaxPool2d: 1-3            kernel_size=2, stride=2, padding=0
Linear: 1-4               in_features=196, out_features=128
Linear: 1-5               in_features=128, out_features=196
ConvTranspose2d: 1-6      channels=16, kernel_size=2, stride=2
ConvTranspose2d: 1-7      channels=16, kernel_size=2, stride=2
==========================================================================
Total params: 51,577
Trainable params: 51,577
Non-trainable params: 0
==========================================================================
```

C.4 Structured Non-IID FEMNIST Dataset

As we explained in Section 4, our newly introduced "Structured Non-IID FEMNIST" is built upon the "balanced" subset of EMNIST dataset [38]. We impose two levels of non-IIDness: i) quantity skew: meaning that the clients will have a varying number of data samples; ii) structured label skew: where we predefined a set of C clusters and clients in different clusters would have uncorrelated sets of labels. The sampling strategy leading to the Structured Non-IID FEMNIST is summarized in the form of pseudo-code in Algorithm 4. In the algorithm, $\mathcal{L}_c$ refers to the set of labels clients in cluster c sample their data from. $\mathcal{C}_c$ denotes the client membership set of cluster c containing the client IDs assigned to the cluster with $|\mathcal{C}_c| = M_c$ and $\sum_c M_c = M$. As such, in this construct, we are grouping clients into a total of C disjoint clusters and the members of each cluster would samples from specific set of labels in $\mathcal{L}_c$. The power law for quantity skew is assumed to follow $\alpha + e^{\beta\, m^\delta}$, where α is a constant denoting the minimum number of samples a client can have and δ is the power exponent constant. Obviously, any reasonable power law can straightforwardly be accommodated here. The algorithm requires, δ and α, as well as the details of the predefined cluster formation and client membership encoded in $\mathcal{L}$ and $\mathcal{C}$. Firstly, the training data is sorted according to the labels per cluster $\mathcal{D}^t = \{\mathcal{D}^t_{\mathcal{L}_1}, \cdots, \mathcal{D}^t_{\mathcal{L}_C}\}$. Next, each cluster and associated clients are processed. Per cluster, depending on the membership size ($M_c := |\mathcal{C}_c|$) the appropriate β is calculated, which will be the same for clusters of the same size. To ensure clients take their samples from almost all the labels allowed in that cluster (denoted by $\mathcal{L}_c$ for cluster $c \in [C]$), a random shuffling is applied to the data samples, and therefore to the associated labels. Finally, within each cluster, a batch of data samples of size $\alpha + e^{\beta^*_c\, m^\delta}$ (following the power low) is assigned to client m in cluster c. More details and the implementation can be found in our GitHub repository shared earlier.

D Mathematical Support

D.1 Accuracy of Clustering

In this section, we provide an approximate upper bound on the taskonomy performance error by considering some simplifying assumptions. Since our approach is founded upon separating client data in latent space, the overall clustering performance for the entire M clients is determined by the autoencoder performance which in turn is affected by several parameters including the number of data samples provided to each client n_m with $\sum_{m\in[M]} n_m = N$, and total number of target classes (labels), L.

We denote the autoencoder error in classifying a single data sample by P_e^{ae}. We also denote the average detection error of the autoencoder over data samples with label l by P_e^l. Besides, we assume that the downstream tasks of individual clients are independent. We then can write

$$P_e^{\text{ae}} = P(y > \gamma), \tag{2}$$

where, γ is a threshold value, and $y \in \mathbb{R}^e$ is the output of the encoder in the latent domain which is assumed to be a multivariate normal distribution with mean μ and standard deviation of σ, $y \sim N(\mu, \sigma^2)$. Let us denote the total number of samples with label l for client m by n_m^l. The client then applies the encoder followed by `kMEANS`. Assuming that all samples corresponding to each label fall within one of these k clusters, the performance of the autoencoder after averaging over samples corresponding to label l can be given by:

$$P_e^l = P(\overline{y} > \gamma), \tag{3}$$

with $\overline{y} \sim N(\mu, \sigma^2/n_m^l)$. In general, $\overline{y}$ would be drawn from a multivariate Gaussian distribution even though here for the sake of simplicity we considered a single-variate one. The error in equation 3 is given by the following Q-function:

$$P_e^l = Q\left(n_m^l \times \mathcal{Q}^{-1}(P_e^{\text{ae}})\right). \tag{4}$$

By employing the Chernoff bound associated with a Q-function [43], the following bound can be given for equation 4:

$$P_e^l \leq \exp\left(-\frac{\left(n_m^l \times \mathcal{Q}^{-1}(P_e^{\text{ae}})\right)^2}{2}\right). \tag{5}$$

By considering our earlier assumption on the independence of classification tasks conducted by the individual clients, the probability of correct classification of the client across all labels will be:

$$1 - P_e^{total} = \prod_{m=1}^{M}\prod_{l=1}^{L}(1 - P_e^l). \tag{6}$$

Accordingly, the total error can be given by:

$$P_e^{total} \approx \sum_{m=1}^{M}\sum_{l=1}^{L} P_e^l. \tag{7}$$

By inserting equation 5 in equation 7, we arrive at:

$$P_e^{total} \lessapprox \sum_{m=1}^{M}\sum_{l=1}^{L} \exp\left(-\frac{\left(n_m^l \times \mathcal{Q}^{-1}(P_e^{\text{ae}})\right)^2}{2}\right). \tag{8}$$

which provides an upper-bound on the overall clustering error. This approximate derivation tells us that by increasing the sample size, the error probability of the encoder vanishes, as desired.

D.2 Convergence of FLT

We prove that under common assumptions, FLT converges. For analyzing the convergence characteristics of the proposed taskonomy-based federated learning approach, the following assumption are in place:

Assumption 1 (Lower boundedness)**.** *All the local cost functions, F_i, are lower bounded:*

$$F_i(w_i) \geq \tilde{\mathcal{F}} > -\infty, \quad \forall w_i \in \mathbb{R}^d. \tag{9}$$

Assumption 2 (Lipschitz continuity)**.** *The function F is L-smooth.*

$$\|F(w_1) - F(w_2)\| \leq L_w \|w_1 - w_2\|, \quad \forall w_1, w_2 \in \mathbb{R}^d. \tag{10}$$

Assumption 3 (Compactness)**.** *Set of local models $\mathcal{W} = \{w_i | i = 1, 2, ..., M\}$ is compact and convex.*

$$\|\nabla_w F_i(w) - \nabla_w F(w)\|^2 \leq \eta^2, \quad \|\nabla_w F_i(w)\|^2 \leq \rho^2, \tag{11}$$

for all $w \in \mathbb{W}$, where η and ρ are some constants.

The FLT optimization problem can be written as the following generalized form:

$$\min_{w_1, w_2, ... w_M \in \mathbb{R}^d} \sum_{m=1}^{M} p_m F_m(w_i), \quad where \;\; w_i = \sum_{j=1}^{M} A_{i,j} w_j, \tag{12}$$

where, A represents the unit-row-sum connectivity matrix and, has the following property:

$$\sum_{j=1}^{M} A_{i,j} = 1, \quad i = 1, 2, ..., M. \tag{13}$$

Note that the term on right side of the equation 12 corresponds to the updates on the server side. This is the part that is handled differently for cluster-based algorithms (e.g. FedSEM, FLT and IFCA). For FedAvg, we have $A = \frac{1}{M} J_M$ where J_M is an $M \times M$ all-ones matrix. In addition, the gradient descent update at the client side is given by

$$w_i^t = w_i^{t-1} - \mu \nabla_{w_i} F_i(w_i^{t-1}), \tag{14}$$

where, μ is step size.

The right-hand term in equation 12 which is performed at the server side to update the model parameters can be written in a matrix form as the following:

$$W^{\texttt{updated}} = AW. \tag{15}$$

Inserting equation 15 in equation 14, we have

$$W^t = A(W^{t-1} - \mu \nabla_W F(W^{t-1})) \tag{16}$$

Following the mathematical framework defined in [44], the optimality gap between the stationary solution of equation 12 is defined as:

$$G_{\mathbb{W}}(W^t) \triangleq \|W^t - W^{t-1}\|^2. \tag{17}$$

Theorem 1 (Convergence of optimality gap)**.** *Under Assumptions 1, 2 and 3, the iterative gradient descent algorithm for problem equation 12 satisfies:*

$$\frac{1}{TM} \sum_{i=1}^{M} \sum_{t=1}^{T} E[G_{\mathbb{W}}(W^t)] \leq \frac{1}{T \times M} \frac{F(W^1) - F(W^T)}{(\frac{1}{2\mu} - \frac{L_W}{2} - \frac{\mu}{2} L_W^2 \|I - A\|^2)}, \tag{18}$$

where, I represents the $M \times M$ identity matrix.

Proof. Since F is L-smooth under Assumption 1, using Lemma 1 in [45], the following holds:

$$F(W^t) \leq F(W^{t-1}) + \langle \nabla_W F(W^{t-1}), W^t - W^{t-1} \rangle + \frac{L_W}{2} \|W^t - W^{t-1}\|^2. \tag{19}$$

The first term of equation 19 is bounded as given by the following lemma:

Lemma 1. *For any $t \geq 0$, the following inequality holds:*

$$\langle \nabla_W F(W^{t-1}), W^t - W^{t-1} \rangle \leq -\frac{1}{\mu} \|W^t - W^{t-1}\| + \langle \nabla_W F(W^{t-1}) - A \nabla_W F(W^{t-1}), W^t - W^{t-1} \rangle. \tag{20}$$

The proof of the above lemma is given in the following:

Proof. Based on the gradient update equation of equation 14, and employing the right-hand term in equation 12, we have:

$$-\frac{w_i^t - w_i^{t-1}}{\mu} = \sum_{j=1}^{M} A_{i,j} \nabla_w F_j(w_j^{t-1}). \tag{21}$$

which leads to:

$$\langle \nabla_w F(W^{t-1}) - A \nabla_w F(W^{t-1}), W^t - W^{t-1} \rangle = \langle \nabla_w F(W^{t-1}) + (W^t - W^{t-1})/\mu, W^t - W^{t-1} \rangle \tag{22}$$

$$\geq \langle \nabla_W F(W^{t-1}), W^t - W^{t-1} \rangle + \frac{1}{\mu} \langle W^t - W^{t-1}, W^t - W^{t-1} \rangle, \tag{23}$$

completing the proof of Lemma 1. □

Continuing on equation 19, by inserting equation 20 in equation 19 and using $\langle x, y \rangle \leq \frac{1}{2c}\|x\|^2 + \frac{c}{2}\|y\|^2$ with $c = \frac{1}{\mu}$, we have

$$F(W^t) - F(W^{t-1}) \leq \left(-\frac{1}{2\mu} + \frac{L_W}{2} \right) \|W^t - W^{t-1}\|^2 + \frac{\mu}{2} \|\nabla_W F(W^{t-1}) - A \nabla_W F(W^{t-1})\|^2, \tag{24}$$

which can be further simplified as:

$$F(W^t) - F(W^{t-1}) \leq \left(-\frac{1}{2\mu} + \frac{L_W}{2} \right) \|W^t - W^{t-1}\|^2 + \frac{\mu}{2} L_W^2 \|I - A\|^2 \|W^t - W^{t-1}\|^2. \tag{25}$$

Finally, taking summation over t from 1 to T in equation 25, we get:

$$\left(\frac{1}{2\mu} - \frac{L_W}{2} - \frac{\mu}{2} L_W^2 \|I - A\|^2 \right) \sum_{t=1}^{T} \|W^t - W^{t-1}\|^2 \leq F(W^1) - F(W^T), \tag{26}$$

which after taking expectation over client models, completes the proof of the theorem. □