COSMOS: Model-Agnostic Personalized Federated Learning with Clustered Server Models and Pseudo-Label-Only Communication

Ben Rachmut[1], Luise Ge[1](✉), Ning Zhang[1], William Yeoh[1], and Yevgeniy Vorobeychik[1]

Washington University in St. Louis g.luise@wustl.edu

Abstract. Federated learning (FL) in heterogeneous environments remains challenging because client models often differ in both architecture and data distribution. While recent approaches attempt to address this challenge through client clustering and knowledge distillation, simultaneously handling architectural and statistical heterogeneity remains difficult. We introduce COSMOS, a model-agnostic framework that enables server-side personalization using only pseudo-label communication. Clients train local models and predict on the public data; the server clusters clients by prediction similarity, trains a cluster-specific model for each group using its own compute, and distills the resulting models back to clients. We provide the first theoretical analysis showing that distillation from the learned cluster models can yield exponential personalization risk contraction, going beyond the convergence-to-stationarity guarantees typically provided in model-agnostic FL. Experiments across benchmarks demonstrate that COSMOS consistently outperforms all model-agnostic FL baselines while remaining competitive with state-of-the-art personalized FL methods. More broadly, our results highlight personalized server-side learning with pseudo-labels as a promising paradigm for scalable and model-agnostic federated learning in highly heterogeneous environments.

Keywords: Federated Learning · Personalized Federated Learning.

1 Introduction

Federated learning (FL) is a distributed training paradigm where clients collaboratively train one or more server-side models without disclosing local data [12, 18, 22]. Motivated by the heterogeneity of client data distributions, a host of personalized FL (PFL) methods has been developed to tailor models to individual clients [1, 3, 16, 29]. However, most existing FL schemes still presume some structural knowledge or compatibility of client model architectures. This imposes a significant practical barrier, as clients may often wish to use whatever model architecture best fits their needs, or make use of proprietary architectures that they would not wish to disclose. Consequently, an important practical need in FL is to be *model agnostic*, allowing clients using whatever models they choose

to simply "plug in" to the FL scheme. The implication of model-agnostic FL is that it disallows any communication about model parameters or gradients.

Despite the clear practical need, the problem of model-agnostic FL remains underexplored, particularly when clients simultaneously differ in both data distributions and model architectures. Existing model-agnostic approaches therefore rely on output-level communication, typically utilizing a shared unlabeled dataset [1,3,16]. Such datasets provide a common reference set on which heterogeneous models can exchange predictive signals without revealing parameters or private data. In many practical deployments, such datasets are readily available through public corpora (e.g., web-scraped images or text), synthetic generation, or institutionally shared benchmark pools. As a result, prediction-based communication has emerged as one of the most practical mechanisms for enabling collaboration across heterogeneous models in FL. Moreover, by using pseudo-labels instead of parameters or gradients, communication efficiency can be improved significantly, which has been widely recognized as a critical concern in FL as the wireless and other end-user connections are typically slower, more expensive and less reliable [26].

To the best of our knowledge, COMET [3] is the closest prior work that targets the crossover. However, COMET has three major limitations. First, the server acts solely as a passive coordinator, neglecting the significant computational resources potentially available at the server side that could be leveraged to facilitate personalization. Second, its reliance on heuristic K-means clustering necessitates prior knowledge of the cluster count K and lacks a formal mechanism for personalization when the underlying client diversity is high. Consequently, COMET's theoretical framework is restricted to standard convergence-to-stationarity for non-convex objectives, providing no formal guarantees regarding personalization performance or risk reduction.

We address all these limitations with COSMOS (**C**lustered **O**utput-based **S**erver **Mo**del**s**). While a high-capacity server has the potential to assist clients, it is highly non-trivial whether the server can effectively learn from the noisy, heterogeneous pseudo-labels provided by the clients in the first place. COSMOS explicitly overcomes this bottleneck through careful algorithmic design. Specifically, COSMOS enables clients to train arbitrary local models on their private data and use them to generate pseudo-labels on a shared unlabeled dataset. The server then performs distance-controlled clustering to group clients with similar data distributions and trains a dedicated teacher model for each cluster. Notably, COSMOS does not require the dataset to match client distributions exactly, only that it provides broad coverage of the input space. As we show in the experiments (Section 6), COSMOS remains effective even when the public dataset constitutes only a small fraction of the overall training data. At the same time, the quality of this public pool remains a genuine limiting factor: when coverage is poor or the public/private distribution shift is severe, both clustering quality and pseudo-label reliability can degrade, so public-data construction is an important practical design choice rather than a free assumption.

Another critical gap in all existing model-agnostic PFL literature is the absence of risk contraction guarantees. To address this, we provide the first end-to-end analysis of personalization risk contraction. To achieve this level of rigor, we leverage standard tools from semi-supervised learning (SSL) theory [14, 31], including expansion-based connectivity and bounded pseudo-label error. They allow us to derive general sufficient conditions for exponential risk contraction without restricting the model class. Importantly, these assumptions are used only for analysis and do not impose constraints on the practical implementation of COSMOS.

Our main contributions are:

1. **Algorithmic Framework.** We present COSMOS, the first model-agnostic PFL framework where the server actively trains cluster-specific models using clients' pseudo-labels.
2. **Theoretical Guarantee.** We establish an end-to-end exponential contraction of personalization risk bounds for COSMOS under sufficient conditions, providing the first general risk contraction guarantee in model-agnostic PFL.
3. **Empirical Evaluation.** We demonstrate that COSMOS not only consistently outperforms existing model-agnostic FL methods but also maintains competitive performance in homogeneous settings, while reducing communication from parameter sharing by 1-2 orders of magnitude.

2 Related Work

Classical Federated Learning. Federated learning was introduced through FedAvg [22], which trains a single global model by aggregating client weight updates. While simple and communication-efficient, FedAvg suffers with non-IID data, motivating methods such as FedProx [18] and SCAFFOLD [12] that stabilize optimization via proximal or control-variance corrections. Nonetheless these methods converge to a single global model and do not offer personalization.

Model-Agnostic, Model-Heterogeneous, and Knowledge-Distillation–Based Federated Learning. Since classical federated learning communicates parameters or their updates, it requires every model on the clients and the server to share the same architecture. To provide greater flexibility, a number of model-heterogeneous approaches, such as communicating instance-level representations as in FedHeNN [21], or abstract class prototypes as in FedProto [30], have been proposed to relax the architectural homogeneity assumption, as have many personalized FL methods (see below). However, while model-agnostic approaches are necessarily model-heterogeneous, most model-heterogeneous methods are *not* model-agnostic, since they still impose some architectural constraints. Additionally, model-agnostic FL is necessarily knowledge-distillation–based FL (KD-FL) [23]. Nevertheless, many KD-FL methods still rely on parameter aggregation at certain stages [2, 19, 25, 34]. To our knowledge, only FedMD [16] and COMET [3] explore purely model-agnostic FL using soft labels, while FedCT [1] relies on hard labels. Recently, the communication efficiency of pseudo-labels

has also been leveraged in federated multi-view clustering (e.g., CeFMC [20]) although its objective is orthogonal to ours.

Personalized Federated Learning (PFL). Almost all practical federated learning settings exhibit statistical heterogeneity, where different clients' local distributions can vary substantially. Personalized federated learning (PFL) addresses this by learning client-adapted models [29]. Existing PFL approaches can be broadly grouped by whether they maintain a single shared server model or a small number of server-side models. In the first group, a single global model is adapted to each client via meta-learning (Per-FedAvg [8]), regularization (pFedMe [6], Ditto [17]), adaptive mixing of local and global models (APFL [5]), representation refinement (FedBABU [24]), or hypernetwork-based parameter generation (pFedHN [27], FedSelect [28]). In the second group, clustered PFL methods explicitly maintain multiple server-side models and assign clients to them, as in IFCA [10], FedGroup [7], AutoCFL [11], and pFedCK [33]. To the best of our knowledge, only COMET is also a model-agnostic PFL approach [3]. However, it requires one to specify the number of clusters K in advance, and applies standard K-means clustering, which is heuristic and not easily amenable to theoretical personalization guarantees.

Theory for Model-Agnostic FL. While theoretical convergence results abound for conventional FL schemes, model-agnostic settings as well as personalization make such results significantly more challenging. The earliest model-agnostic FL approach, FedMD [16] does not provide any theoretical guarantees. A recent FedCT method [1] requires an oversimplified assumption directly that the training algorithms *always* yield monotone increasing accuracy to achieve convergence. The theoretical analysis for COMET [3], on the other hand, requires linear models and a Gaussian data distribution to obtain generalization results. Thus, there are no general sufficient conditions on risk bound contraction for model-agnostic personalized FL. We bridge this gap by adopting the analysis tools from the semi-supervised learning literature [31].

3 Model

We consider a federated learning scenario for M-class classification with N clients and a server. Each client $i \in [N]$ has a private labeled dataset $D_i = \{(x_{ij}, y(x_{ij}))\}_j$ with x_{ij} drawn i.i.d. from its local distribution $\mathcal{D}_i$ over the input space $\mathcal{X}$, and $y(x_{ij})$ the true label of x_{ij}. We assume that each $\mathcal{D}_i$ admits a density function $p_i(x)$. A client i trains a model $f_i : \mathcal{X} \to [0,1]^M$ with $\|f_i(x)\|_1 = 1$ from a hypothesis class $\mathcal{H}_i$ that can be distinct for each i, representing, for example, distinct neural network architectures for different clients. A server, in turn, has a hypothesis class $\mathcal{H}_S$ and can train a *collection* of models $H = \{h_1, \ldots, h_K\} \subset \mathcal{H}_S$, where $h_k : \mathcal{X} \to [0,1]^M$ and $\|h_k(x)\|_1 = 1$ for each k. In our setting, the value of K is obtained *endogenously as part of the training procedure*. Furthermore, let $\pi : [N] \to [K]$ be a mapping (also obtained during training) which assigns each client i to a corresponding server model $h_{\pi(i)}$.

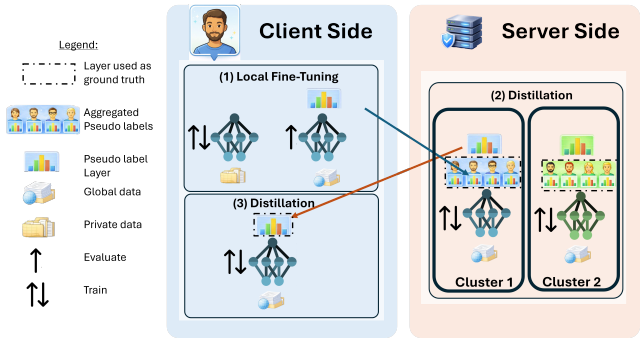

Fig. 1. Overview of the four steps in the COSMOS workflow.

Finally, following prior work on PFL, we assume availability of an *unlabeled public dataset* $U = \{x_j\}$ of size $n = |U|$, with each x_j drawn i.i.d. from a global distribution $\mathcal{Q}$ over $\mathcal{X}$. We assume that $\mathcal{Q}$ admits a density function $q(x)$. It is not difficult to obtain unlabeled datasets of this kind, for example, scraping (open-license) images or text from the internet, or generating it synthetically.

To formalize the learning objective, we introduce some additional notation. Let g be a classification model with outputs a distribution over M classes (i.e., pseudo-labels). We use $A \circ g(x) = \arg\max_{m \in [M]} [g(x)]_m$ to denote the predicted class (i.e., the class with the highest probability under g). Further, let $R_{\mathcal{D}}(g) = \mathbb{E}_{x \sim \mathcal{D}}[\ell_{0-1}\{A \circ g(x), y(x)\}]$ denote the risk (probability of a mistake) of g under distribution $\mathcal{D}$, where ℓ_{0-1} is the $0-1$ loss, and $\mathrm{Err}_{x \in D}(g) = \frac{1}{|D|} \sum_x \ell_{0-1}\{A \circ g(x), y(x)\}$ denote the hard label prediction error over a dataset D.

Learning Objective: Our goal is to train a collection $\{h_k\}$ of K personalized server models, along with a client-to-model mapping π, which minimizes the *personalization risk*, defined as the average risk over the N clients: $R^{\dagger}(h_1, \ldots, h_K) = \frac{1}{N} \sum_{i=1}^{N} R_{\mathcal{D}_i}(h_{\pi(i)})$.

4 Algorithmic Approach

At the high level, the proposed COSMOS framework involves iteratively fine-tuning the client and server models using pseudo-labels collected from one another. As shown in Figure 1, clients generate pseudo-labels for global data using locally trained models, which the server clusters and aggregates to train cluster-specific models. The resulting pseudo-labels are iteratively returned to clients for local model refinement. We enable personalization by creating and training a small collection H of K server models, with each server model h_k specialized to a subset of similar clients obtained through clustering. More precisely, COSMOS proceeds in two phases: 1) the *pre-training and clustering (PTC)* phase and 2) the *iterative federated fine-tuning (IFFT)* phase.

4.1 The Pre-Training and Clustering Phase

The pre-training and clustering phase involves four steps: 1) local pre-training, 2) clustering, 3) server-side distillation, and 4) client-side distillation.

Step 1: Local Pre-Training. We pre-train each client i's model f_i locally on D_i for E_0 epochs using conventional (e.g., cross-entropy) loss. We refer to the resulting client models as $f_i^{(1,1)}$, where the first superscript refers to the time step ($t = 1$) and the second means that the client models are trained (or fine-tuned, in the IFFT phase) on *local* data (as opposed to global data U with server-provided pseudo-labels, as in Step 4 below). Each client i then generates pseudo-labels over the global set U, $f_i^{(1,1)}(U)$, and sends these to the server.

Step 2: Client Clustering. Upon receiving pseudo-labels $f_i^{(1,1)}(U)$ from all clients i, the server proceeds to cluster the clients into K clusters $\{\mathcal{G}_k\}_{k=1}^K$ based on pseudo-label similarity. Our goal is to find the minimum number of clusters K with respect to a given distance function. However, this problem is NP-hard and the best known approximation has computational complexity exponential in the data dimension [9]. On the other hand, while numerous heuristic clustering approaches have been proposed, they cannot be easily used to provide *risk convergence* guarantees.

To address this challenge, we propose the following greedy approach, which *is* amenable to theoretical analysis (see Section 5.1). We define the distance between two clients i, j as their pseudolabels' ℓ_1 distance, i.e. $d^{(t)}(i,j) = |f_i^{(t,1)}(U) - f_j^{(t,1)}(U)|_1$. Define the set $N_i(B_0, C)$ of B_0-*close neighbors* of a client i among clients in C as $N_i(B_0, C) = \{j | j \in C \setminus \{i\}, d^{(1)}(i,j) \leq B_0\}$. At the high level, our algorithm greedily selects a client with the most B_0-close neighbors among those not previously selected, and defines a new cluster associated with this client and its neighbors, proceeding until all clients belong to some cluster. This is made more precise in Algorithm 1. The resulting clustering scheme induces a cluster assignment function π which maps each client i to a cluster k. Notably, *we need not know the number of clusters upfront*, as this is endogenous to our clustering approach. While B_0 can be treated as another hyperparameter for clustering, this hyperparameter is theoretically grounded: it is directly connected to the personalization guarantees of our framework. In practice, this means COSMOS is not hyperparameter-free: although K is not supplied directly, the threshold B_0 must still be tuned, for example by validation performance or a communication/compute budget.

Step 3: Server-Side Distillation. Our next step is to train the server-side model in each cluster k based on the pseudolabels received from all clients. Specifically, fix a cluster $\mathcal{G}_k$ and a datapoint $x \in U$. The server update for each cluster model $h_k^{(1)}$ (where the superscript references iteration) and $x \in U$ then uses the aggregate pseudo-labels over all clients in this cluster which we define as $\bar{f}_k^{(1)}(x) = \frac{1}{|\mathcal{G}_k|} \sum_{i \in \mathcal{G}_k} f_i^{(1,1)}(x)$ for clients $i \in \mathcal{G}_k$, i.e., the average of the pseudo-labels of the clients in the k-th cluster. Finally, the server performs gradient-based training for each cluster-specific model k to minimize the follow-

Algorithm 1: Greedy Clustering.

Input: Clients $1, \ldots, N$ with associated hard label vectors $\{y_i\}$; clustering parameter B_0.
Output: The number of clusters K and the associated collection of client clusters $G = \{\mathcal{G}_k\}_{k=1}^{K}$.
Initialization: $C = [1, \ldots, N]$, $G = \{\emptyset\}$, $K = 1$
while $C \neq \emptyset$ **do**
  $i^* \in \arg\max_{i \in C} |N_i(B_0, C)|$
  $\mathcal{G}_K = \{i^* \cup N_{i^*}(B_0, C)\}$
  $G = G \cup \mathcal{G}_K$; $C = C \setminus \mathcal{G}_K$; $K = K + 1$

ing objective:

$$\hat{J}(h_k^{(1)}; \ell) = \sum_{x \in U} \Big[\ell(h_k(x), \bar{f}_k^{(1)}(x)) + \lambda \, r_{\mathcal{B}}(h_k^{(1)}; \ell)(x) \Big], \tag{1}$$

where $\ell(\cdot, \cdot)$ is a loss function and $r_{\mathcal{B}}(\cdot, \ell)$ an input-specific regularization term. The role of the regularization term in our setting is to train the models which are locally smooth (related to the consistency and robustness properties central to our theoretical analysis in Section 5). Specifically, let $\mathcal{T}$ denote a set of permissible data augmentations (e.g., image translations and rotations) and fix d to be a small radius capturing minor perturbations around augmented data points (in particular, we assume that d is significantly smaller than the typical norm of x). For an input $x \in \mathcal{X}$, define its *transformation ball* as: $\mathcal{B}(x) = \{x' \in \mathcal{X} : \exists T \in \mathcal{T} \text{ s.t. } \|x' - T(x)\| \leq d\}$. Next, let the neighborhood $\mathcal{N}(x)$ of a point x be the set of inputs whose transformation balls intersect with $\mathcal{B}(x)$: $\mathcal{N}(x) = \{x' : \mathcal{B}(x) \cap \mathcal{B}(x') \neq \emptyset\}$. Let $S \subset \mathcal{N}(x)$ be a finite sample of the neighborhood of x (e.g., obtained by rejection sampling). We now define a regularization term for any model g as $r_{\mathcal{B}}(g; \ell)(x) := \sum_{x' \in S} \big\{ \ell(g(x'), g(x)) \big\}$. We use the same loss function for the supervised term and regularized term, but allowing different loss functions is also admissible. We motivate this form of regularization more precisely in Section 5. This server-side stage is also the main additional systems cost of COSMOS relative to passive-server methods such as COMET: the total server work scales with the number of clusters, the size of the public pool, and the chosen server architecture, although the cluster models can be trained in parallel when compute is available.

After gradient-based training for each server-side model $h_k^{(1)}$ using the objective (1), we send the resulting pseudo-labels $h_k^{(1)}(U)$ to clients in cluster $\mathcal{G}_k$.

Step 4: Client-Side Distillation. Finally, each client i fine-tunes its local model $f_i^{(1,2)}$, where the second superscript refers to the second round of local fine-tuning based on the pseudo-labels $h_{\pi(i)}^{(1)}(U)$. We use the following objective:

$$\hat{F}(f_i^{(1,2)}; \ell) = \sum_{x \in U} w_i(x) \Big[\ell(f_i^{(1,2)}(x), h_{\pi(i)}^{(1)}(x)) + \lambda \, r_{\mathcal{B}}(f_i^{(1,2)}; \ell)(x) \Big], \tag{2}$$

where $w_i(x)$ are the client-specific weights of data in U that allow for importance sampling techniques (see Section 5). In practice, we find that setting $w_i(x) = 1$ for all i and x is effective in practice, and avoids the need to make assumptions about the local and global distribution differences.

4.2 The Iterative Federated Fine-Tuning Phase

After the PTC phase, we enter an iterative FL phase in which we alternate 1) local fine-tuning, 2) server-side distillation, and 3) client-side distillation over a fixed number of iterations. We now describe each step for a fixed iteration t.

Step 1: Local Fine-Tuning. The PTC phase effectively serves as iteration $t = 1$. In any IFFT iteration $t \geq 2$, the local fine-tuning step for client i starts from the model $f_i^{(t-1,2)}$ (obtained by fine-tuning on the public data U annotated with the server's pseudo-labels, as described in Step 3 below), and performs supervised training on its labeled local dataset $D_i = \{(x, y)\}$ to obtain $f_i^{(t,1)}$. Concretely, as before, client i approximately minimizes the empirical risk with a calibrated surrogate loss ℓ such as cross-entropy loss $\hat{R}_{\ell,D_i}(f) = \frac{1}{|D_i|}\sum_{(x,y)\in D_i} \ell\big(f(x), y\big)$. For the contraction analysis, we explicitly assume this supervised update is *classification-safe*: with high probability over the local sample and optimization noise, it does not increase the true 0–1 risk on $\mathcal{D}_i$ beyond the generalization slack. After fine-tuning, the pseudo-labels $f_i^{(t,1)}(U)$ are then sent to the server to fine-tune the model $h^t_{\pi(i)}$.

Step 2: Client-Side Distillation. In iteration t, each server model $k \in [1, \ldots, K]$ is fine-tuned on the pseudo-labels $f_i^{(t,1)}(U)$ received from clients $i \in \mathcal{G}_k$ in a way that closely mirrors Step 3 of the PTC phase. That is, we define $\bar{f}_k^{(t)}(x) = \frac{1}{|\mathcal{G}_k|}\sum_{i\in\mathcal{G}_k} f_i^{(t,1)}(x)$ for $i \in \mathcal{G}_k$. The server model $h_k^{(t)}$ is then fine-tuned using the objective (1), with $h_k^{(1)}$ replaced by $h_k^{(t)}$ and $\bar{f}_k^{(1)}(x)$ replaced by $\bar{f}_k^{(t)}(x)$. The server then sends pseudo-labels $h_k^{(t)}(U)$ to clients $i \in \mathcal{G}_k$ for all k.

Step 3: Client-Side Distillation. Finally, we fine-tune each client i from the pseudo-labels $h_{\pi(i)}^{(t)}(U)$ using the objective from Equation 2 (as in Step 4 of PTC), in which we replace $f_i^{(1,2)}$ with $f_i^{(t,2)}$ and $h_{\pi(i)}^{(1)}$ with $h_{\pi(i)}^{(t)}$.

5 Personalization Guarantees

In this section, we analyze the population risk of COSMOS with respect to the client distributions $\{\mathcal{D}_i\}_{i=1}^N$. We provide such guarantees both for the individual client models, as well as for the overall *personalization risk* on the server. This analysis entails several technical challenges. First, as clustering is NP-hard, heuristic approaches are typically used, and it is therefore not evident how to achieve convergence guarantees for personalization risk for algorithms that leverage clustering (as COSMOS does). Second, it is clear that arbitrary distributions

$\mathcal{D}_i$ (of the client data) and $\mathcal{Q}$ (of global data) cannot achieve convergence. For example, if $\mathcal{Q}$ puts all mass on a single datapoint, the clustering step will fail. Prior approaches deal with this issue by making strong distributional assumptions, such as assuming a Gaussian data distribution [3]. We aim to obtain *general* sufficient conditions on the data distributions. Third, we wish to make minimal assumptions on the nature of client and server models, unlike prior approaches that require linearity for generalization bounds [3, 27].

Notation: Before diving into the technical details, we define some useful notation. For a client i, let $\mathcal{S}_i = \text{supp}(\mathcal{D}_i) \subseteq \mathcal{X}$ denote the support of i's local data distribution and $\mathcal{S}_i^c = \mathcal{X} \setminus \mathcal{S}_i$ be its complement. We write $U_i = U \cap \mathcal{S}_i$ for the client-supported portion of public data and $U_i^c = U \setminus U_i$ for its complement. Let $\mathcal{Q}(\mathcal{S}_i)$ denote the probability of $\mathcal{S}_i$ under $\mathcal{Q}$. Let $\mathcal{Q}_i := \mathcal{Q}(\cdot \mid \mathcal{S}_i)$ denote the restriction of $\mathcal{Q}$ to $\mathcal{S}_i$, with density $q_i(x) = q(x)/\mathcal{Q}(\mathcal{S}_i)$. Let $\mathcal{Q}_i^c := \mathcal{Q}(\cdot \mid \mathcal{S}_i^c)$. For any functions g and g', and input x, define $G(g, g', \ell) = \ell(g, g') + \lambda r_{\mathcal{B}}(g; \ell)$. Then, we can write the server-side cluster k's objective $\hat{J}(h_k) = \sum_{x \in U} G(h_k, \bar{f}_k, \ell)(x)$, where h_k is the server-side model and $\bar{f}_k$ the aggregate model over clients in the kth cluster. Similarly, we can write $\hat{F}(f_i; \ell) = \sum_{x \in U} w_i(x) G(f_i, h_{\pi(i)}, \ell)(x)$ for a client i's model f_i. It will also be useful to define $\hat{J}_i(h_k; \ell) = \sum_{x \in U_i} G(h_k, \bar{f}_k, \ell)(x)$ and $\hat{J}_i^c(h_k; \ell) = \sum_{x \in U_i^c} G(h_k, \bar{f}_k, \ell)(x)$ for the server, and similarly, $\hat{F}_i(f_i; \ell) = \sum_{x \in U_i} w_i(x) G(f_i, h_{\pi(i)}, \ell)(x)$ and $\hat{F}_i^c(f_i; \ell) = \sum_{x \in U_i^c} w_i(x) G(f_i, h_{\pi(i)}, \ell)(x)$ for each client i. $\hat{J}_i$ and $\hat{F}_i$ are the server and client objectives with respect to the public data U_i restricted to support of i's data distribution, whereas $\hat{J}_i^c$ and $\hat{F}_i^c$ are these objectives on the portion of U outside client i's support. Since $U = U_i \cup U_i^c$ for each i, we can also note that $\hat{J}(h_k; \ell) = \hat{J}_i(h_k; \ell) + \hat{J}_i^c(h_k; \ell)$ and $\hat{F}(f_i; \ell) = \hat{F}_i(f_i; \ell) + \hat{F}_i^c(f_i; \ell)$.

Moreover, insofar as our interest is in convergence (in terms of *true* risk), the objectives for the clients and server defined above are only estimates thereof. The true objective of each server-side cluster is $J(h_k; \ell) = \mathbb{E}_{x \sim \mathcal{Q}}[G(h_k, \bar{f}_k, \ell)(x)]$, with $J_i(h_k; \ell) = \mathcal{Q}(\mathcal{S}_i) \mathbb{E}_{x \sim \mathcal{Q}_i}[G(h_k, \bar{f}_k, \ell)(x)]$; $J_i^c(h_k) = \mathcal{Q}(\mathcal{S}_i^c) \mathbb{E}_{x \sim \mathcal{Q}_i^c}[G(h_k, \bar{f}_k, \ell)(x)]$; and we define $F(f_i; \ell)$, $F_i(f_i; \ell)$, and $F_i^c(f_i; \ell)$ analogously for each client i. While cumbersome, the key consideration that this notation enables us to deal with is how well-behaved the objective functions of the server and clients are on the supported and unsupported parts of the public data.

5.1 Clustering

Our risk analysis relies on a controlled within-cluster pseudo-label distance B over the shared unlabeled set U. In particular, the greedy clustering procedure in Section 4.1 enforces this property in the first round by constructing clusters with threshold B_0, where $B_0 \leq B$. In subsequent rounds, bounded in-cluster disagreement is expected to continue to hold because all clients are anchored to the same public set and distill from the same cluster-level teacher in each round. This condition is the mechanism that lets us control how much a cluster-level aggregation step can degrade client i's pseudo-labels, which leads to Lemma 1.

Without an explicit bound, similarity-based clustering may work well empirically but offers no convergence guarantees: aggregation within a cluster can introduce uncontrolled label error.

We additionally require clients to have confidence margin on U_i to prevent small within-cluster discrepancies from flipping argmax labels. This is reasonable because $f_i^{(t,1)}$ is obtained by fine-tuning on client i's labeled data, so its predictions on the in-support public subset U_i are expected to be sufficiently decisive. Formally, for any $g(x)$ mapping inputs to a distribution over labels $[M]$, define the margin $\Delta_g(x) = g(x)_{(1)} - g(x)_{(2)}$, where $g(x)_{(m)}$ denotes the m-th largest coordinate of $g(x)$.

Condition 1 *For each iteration t and cluster k, the within-cluster pseudo-label distance is bounded:* $\max_{i,j\in\mathcal{G}_k} d^{(t)}(i,j) \le B$. *Moreover, for each client i there exists $\gamma > 0$ such that for every $x \in U_i$, $\Delta_{f_i^{(t,1)}}(x) \ge \gamma$.*

Lemma 1. *For each client i,* $\mathrm{Err}_{x\in U_i}(\bar{f}_k^{(t)}) \le \mathrm{Err}_{x\in U_i}(f_i^{(t,1)}) + \frac{2B}{\gamma\,|U_i|}$.

5.2 Data and Objective Conditions

Our next challenge comes from the server's reliance on the public unlabeled pool U: since it never observes clients' labeled data, any guarantee must ensure that the distribution Q induced by U provides adequate coverage of each local distribution $\mathcal{D}_i$. In particular, a key requirement is that the public distribution Q does not under-cover nor over-concentrate on regions where client i has support. We formalize this as the following condition.

Condition 2 (Distributional coverage) *For each client i with local data distribution $\mathcal{D}_i$ that has density p_i, $w_{i,1} = \sup_{x\in\mathcal{S}_i} \frac{p_i(x)}{q_i(x)}$ and $w_{i,2} = \sup_{x\in\mathcal{S}_i} \frac{q_i(x)}{p_i(x)}$ are bounded.*

We note that we can in principle *construct* U to be sufficiently diverse, or define a sampling distribution $\mathcal{Q}$ to have a strictly positive density over $\mathcal{X}$, such as a Gaussian distribution. Then, if $\mathcal{X}$ is bounded (for example, $\mathcal{X} = [0,1]^m$, as for normalized image data), the condition will hold for reasonable $\mathcal{D}_i$ (since the sup is over its support).

Furthermore, we must ensure convexity for a well-conditioned optimization landscape, and that data points outside the support of each client's distribution do not excessively interfere with learning from client-supported data, where pseudo-labels are expected to be more accurate. To capture this, we require key data-dependent regularity conditions on the server and client objectives, $J, \hat{J}, F$, and $\hat{F}$, captured by the following definition.

Definition 1 (Locally well-conditioned Objective). *We say a differentiable function $\Phi(\theta;\ell)$ is* (μ, L)-locally well-conditioned *if*

1. *Φ_i is μ-strongly convex: $\nabla^2\Phi_i(\theta) \succeq \mu I$ for $\mu > 0$.*

2. $\nabla\Phi_i^c$ *is* L_2*-Lipschitz continuous:* $\|\nabla\Phi_i^c(\theta) - \nabla\Phi_i^c(\theta')\| \leq L\|\theta - \theta'\|\ \forall\theta, \theta'$ *for some* $0 < L < \mu$.
3. $\|\nabla\hat{\Phi}_i^c(\theta^*)\| \leq \tau$ *where* $\theta^* = \arg\min_\theta \hat{\Phi}_i(\theta)$.

In the following, we note that the functions F, $\hat{F}$, J, and $\hat{J}$ are, effectively, functions of *parameters* of the models they are constructed around, whether these are the server-side models h_k or the client-side models f_i.

Condition 3 *For all clients* i *and COSMOS iterations* t*:*

1. $F(\theta_i^{(t)}; \ell)$, $\hat{F}(\theta_i^{(t)}; \ell)$, $J(\theta_k^{(t)}; \ell)$, *and* $\hat{J}(\theta_k^{(t)}; \ell)$ *are locally well-conditioned, and*
2. *The pseudolabels generated by any pairs of parameters* θ, θ' *from the same hypothesis class are* L_1*-Lipschitz continuous:* $\sup_{x\in\mathcal{X}} \left\|\theta(x) - \theta'(x)\right\|_\infty \leq L_1 \|\theta - \theta'\|$.

The final regularity condition bounds how often inputs fall arbitrarily close to the decision boundary, so that a small change in the objective does not frequently flip the predicted label. Concretely, we adopt a standard Tsybakov-style margin condition for the optimal classifiers.

Condition 4 (Confidence margin condition) *For each client* i*, let* $f_i^* \in \arg\min_{g\in\mathcal{H}_i} F_i(g; \ell)$ *and let* $h_i^* \in \arg\min_{g\in\mathcal{H}_S} J_i(g; \ell)$*. Then for all* i*, there exist constants* $C > 0$ *and* $\alpha > 0$ *such that* $\forall t \geq 0$, $\Pr\left(\Delta_{f_i^*}(x) \leq t\right) \leq C\, t^\alpha$ *and* $\Pr\left(\Delta_{h_i^*}(x) \leq t\right) \leq C\, t^\alpha$.

5.3 Pseudolabel and Label Conditions

While our clustering controls the additional error introduced by aggregation, and our data/objective conditions support stable gradient-based optimization, a purely pseudo-label-based method still requires additional structure to be provably effective. In particular, the pseudo-labels must carry nontrivial information about the ground truth, and the ground truth labels should satisfy some structural connectivity so that the learner can generalize beyond the pseudo-labeled points rather than merely memorizing arbitrary functions. These conditions are what enable learning from pseudo-labels, inducing the weak-to-strong generalization we observe at the server's end [14].

Specifically, we adopt the expansion-and-robustness framework of [31], which expresses label connectivity via input-space transformations $\mathcal{T}$ and associated neighborhoods $\mathcal{N}(x)$ of a given input x (see Section 4.1, Step 3). For a subset $V \subseteq \mathcal{X}$, define $\mathcal{N}(V)$ as the union of neighborhoods of its points: $\mathcal{N}(V) = \bigcup_{x\in V} \mathcal{N}(x)$. Expansion ensures that if a set V contains a small fraction of a given class, then closing V under the neighborhood operator $\mathcal{N}(\cdot)$ captures a larger fraction of that same class.

Definition 2 ((b, c)-expansion). *A distribution* $\mathcal{D}$ *satisfies* (b, c)*-expansion if for every class* m *and* $\forall V \subseteq \mathcal{X}$ *with* $\Pr_{x\sim\mathcal{D}|x\in V}[y(x) = m] \leq b$*, we have* $\Pr_{x\sim\mathcal{D}|x\in\mathcal{N}(V)}[y(x) = m] \geq \min\{c \Pr_{x\sim\mathcal{D}}[y(x) = m \mid c \in V], 1\}$.

To quantify the probability that pseudo-labelers make mistakes for any client i and iteration t, define

$$\bar{b} = \sup_{t,i} \max\Big\{ \mathcal{Q}_i\Big(\mathcal{M}(h^{(t)}_{\pi(i)})\Big),\ \mathcal{Q}_i\Big(\mathcal{M}(\bar{f}^{(t)}_{\pi(i)})\Big),\ \mathcal{D}_i\Big(\mathcal{M}(h^{(t)}_{\pi(i)})\Big),\ \mathcal{D}_i\Big(\mathcal{M}(\bar{f}^{(t)}_{\pi(i)})\Big)\Big\},$$

where $\mathcal{M}(f) := \{x \in \mathcal{X} : A \circ f(x) \neq y(x)\}$.

The following natural condition requires that the probability $\bar{b}$ of pseudo-labeling errors is not too large, as well as inputs with a given class are sufficiently well-connected.

Condition 5 (Effective pseudo-labelers) $\bar{b} \leq \frac{1}{3}$.

Complementing expansion, robustness formalizes the idea that inputs close in the input space share the same labels. We characterize this property in terms of robustness loss.

Definition 3. *For a function f and distribution $\mathcal{D}$,* robustness loss *is the fraction of examples that are not robust to input transformations:*

$$R_{\mathcal{B}}(f, \mathcal{D}) \ = \ \mathbb{E}_{x\sim\mathcal{D}}\Big[\mathbf{1}\big\{\exists\, x' \in \mathcal{N}(x)\colon f(x') \neq f(x)\big\}\Big].$$

Condition 6 (Label connectivity) *For every client i and true classifier $y(x)$, $\mathcal{Q}_i$ and $\mathcal{D}_i$ satisfy $(\bar{b}, \bar{c})$-expansion for some $\bar{c} > 3$ and $\max\{R_{\mathcal{B}}(y, \mathcal{Q}_i), R_{\mathcal{B}}(y, \mathcal{D}_i)\} \leq \rho$.*

5.4 Risk Contraction

We now put these tools together to provide bounds for the client-side and server-side risk. Proofs are omitted for space. Define $c = \min\Big\{\bar{c},\ \frac{1}{\bar{b}}\Big\}$. Suppose that 1) $c > 2w_1w_2 + 1$, where $w_1 = \max_i\{w_{i,1}\}$ and $w_2 = \max_i\{w_{i,2}\}$, 2) $\lambda = \frac{2c}{c+1}$, 3) Conditions 1–6 hold, and 4) each supervised local update is classification-safe: with high probability, the update from $f_i^{(t-1,2)}$ to $f_i^{(t,1)}$ does not increase the true 0–1 risk on $\mathcal{D}_i$ beyond the lower-order generalization slack.

Theorem 1. *For any client i at any iteration t, with probability at least $1-\delta$, its server model $h^{(t)}_{\pi(i)}$'s risk is contracting: $R_{\mathcal{D}_i}(h^{(t)}_{\pi(i)}) \ \leq \ \kappa_1\, R_{\mathcal{D}_i}(f_i^{(t,1)}) + \tilde{O}(n^{-1/2})$, where $\kappa_1 = \frac{2w_1w_2}{c-1} < 1$ and the lower order term hides constants and poly-log terms in n.*

Theorem 2. *Let $\kappa_2 = \frac{4w_1w_2}{(c-1)^2} < 1$. For each client i and iteration t, with probability at least $1-\delta$, $R_{\mathcal{D}_i}\Big(f_i^{(t,2)}\Big) \ \leq \ \kappa_2\, R_{\mathcal{D}_i}\Big(f_i^{(t-1,2)}\Big) + \ \tilde{O}\Big(\sqrt{\frac{\log(1/\delta)}{n}}\Big)$.*

Applying a union bound, we obtain a contraction of the personalization risk bound at an exponential rate.

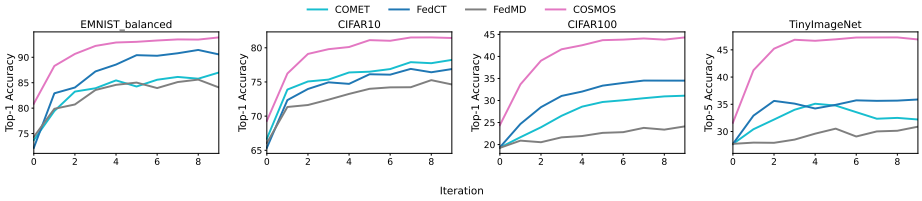

Fig. 2. Comparison of COSMOS and model-agnostic baselines on four benchmarks. Client models consist of MobileNet and SqueezeNet architectures. Curves show mean client accuracy over rounds (Top-1 for CIFAR-10/100 and EMNIST; Top-5 for Tiny ImageNet).

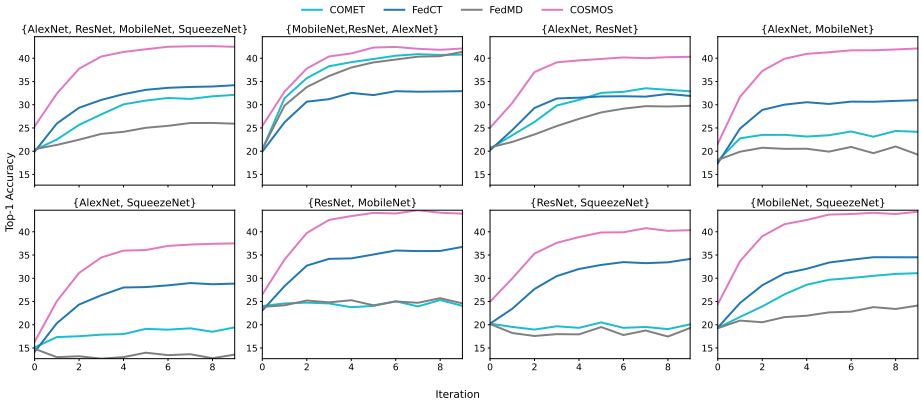

Fig. 3. Performance of COSMOS and heterogeneous-model baselines on CIFAR-100 under different mixtures of client architectures. Each subfigure corresponds to a distinct architecture combination.

Corollary 1 (Personalization Risk Bound). *With probability at least* $1-\delta$ *personalization risk after* T *iterations is*

$$R^{\dagger}\left(h_1^{(T)},\ldots,h_K^{(T)}\right) \leq \frac{1}{N}\sum_{i=1}^{N}\kappa_2^T R_{\mathcal{D}_i}\left(f_i^{(1,1)}\right) + \tilde{O}\left(\sqrt{\frac{\log(T/\delta)}{n}}\right).$$

Moreover, for sufficiently large n and T, personalization risk drops below the risk from local training alone.

6 Experiments

Experiment Setup: We construct client datasets using Dirichlet non-IID partitioning with concentration parameter α [32]. Specifically, the label space is partitioned into five disjoint groups, each containing 20% of the classes, and

clients are assigned to one group. For each class, its samples are distributed among the clients within the same group according to a Dirichlet draw. To introduce limited cross-group exposure and better reflect realistic data sharing, a fraction of each client's local samples (10%) is pooled, randomly shuffled, and uniformly redistributed across all clients. Our unlabeled dataset U consists of 20% of all training data. We use accuracy as the efficacy measure.

We selected client architectures among 4 options: AlexNet, ResNet18, MobileNet, and SqueezeNet. We use VGG16 for the server. Our focus is on the model-agnostic setup, where we compare COSMOS with the only three truly model-agnostic baselines (all others require knowledge of client model architecture; see Section 2): FedMD [16], FedCT [1], and COMET [3].

We use four image classification benchmarks: EMNIST-balanced [4], CIFAR-10 and CIFAR-100 [13], and Tiny ImageNet [15]. The main experiments use 25 clients over 10 communication rounds, using Dirichlet non-IID sampling ($\alpha = 5$). Each data point in the figures reflects the *mean client accuracy* over three random seeds. Because all compared model-agnostic baselines exchange predictions on the shared pool, communication cost is naturally tied to the size of U and the label dimension; even so, prediction exchange remains far cheaper than parameter sharing in our setting. For example, the per-client transmission cost is 0.38 MB for pseudo-labels versus 113.36 MB for parameter transfer on CIFAR-10, and 3.81 MB versus 114.76 MB on CIFAR-100.

Results: Figure 2 presents the main results across all four benchmarks in the model-agnostic setting with $\alpha = 5$. In all cases, COSMOS outperforms all model-agnostic baselines by a large margin. Moreover, its advantage increases with increasing task complexity (from EMNIST to TinyImageNet). This trend is stable across the additional heterogeneity settings $\alpha \in \{1, 100\}$: COSMOS remains strongest on the harder benchmarks and does not lose its advantage when heterogeneity is either increased or relaxed.

Figure 3 evaluates COSMOS under different mixtures of client architectures. Across all combinations, COSMOS remains robust to heterogeneity in client capabilities, including settings with a high proportion of weaker models (AlexNet), maintaining substantial performance edge over baselines. These gains are not free: COSMOS improves further when the shared public pool is larger and the server model is stronger, which is consistent with the theoretical coverage assumptions and with the additional computation required to train cluster-specific teachers.

Finally, we analyzed the sensitivity of COSMOS to key design choices and hyperparameters (including temperature T and weight of the regularization term λ). We find that COSMOS is quite robust to small changes in λ, with $T = 1$ and $\lambda = 5$ yielding the best performance.

7 Conclusion

In this work, we introduce COSMOS, a framework that addresses a critical bottleneck in federated learning: achieving high-performance personalization while

remaining fully model-agnostic. By allowing clients to utilize arbitrary (even proprietary) architectures through the communication of predictions on a shared unlabeled pool, COSMOS removes the structural barriers that have historically limited the deployment of PFL in diverse, real-world ecosystems. We establish a general end-to-end theoretical analysis of risk (generalization) bound contraction of our framework that significantly generalizes past theoretical results in this setting. We also validate the algorithm's effectiveness against state-of-the-art baselines across multiple benchmarks. Current experiments are limited to image classification, and a key next step is to study more severe public/private distribution shift as well as additional modalities and decentralized variants of the framework.

Acknowledgements

This work was supported in part by the National Science Foundation (IIS-2214141, CCF-2403758), Army Research Office (W911NF-25-1-0059), and Office of Naval Research (N00014-24-1-2663).

References

1. Abourayya, A., Kleesiek, J., Rao, K., Ayday, E., Rao, B., Webb, G.I., Kamp, M.: Little Is Enough: Boosting Privacy by Sharing Only Hard Labels in Federated Semi-Supervised Learning. In: Proceedings of the AAAI Conference on Artificial Intelligence (2025)
2. Afonin, A., Karimireddy, S.P.: Towards Model Agnostic Federated Learning Using Knowledge Distillation. In: International Conference on Learning Representations (2022)
3. Cho, Y.J., Wang, J., Chirvolu, T., Joshi, G.: Communication-Efficient and Model-Heterogeneous Personalized Federated Learning via Clustered Knowledge Transfer. IEEE Journal of Selected Topics in Signal Processing (2023)
4. Cohen, G., Afshar, S., Tapson, J., Van Schaik, A.: EMNIST: Extending MNIST to Handwritten Letters. In: International Joint Conference on Neural Networks (IJCNN) (2017)
5. Deng, Y., Kamani, M.M., Mahdavi, M.: Adaptive Personalized Federated Learning. arXiv preprint arXiv:2003.13461 (2020)
6. Dinh, C.T., Tran, N., Nguyen, J.: Personalized Federated Learning With Moreau Envelopes. Advances in Neural Information Processing Systems (2020)
7. Duan, M., Liu, D., Ji, X., Liu, R., Liang, L., Chen, X., Tan, Y.: Fedgroup: Efficient federated learning via decomposed similarity-based clustering. In: IEEE International Conference on Parallel & Distributed Processing with Applications (ISPA/BDCloud/SocialCom/SustainCom) (2021)
8. Fallah, A., Mokhtari, A., Ozdaglar, A.: Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. In: Advances in Neural Information Processing Systems (2020)
9. Ge, L., Lanier, M., Sarkar, A., Guresti, B., Vorobeychik, Y., Zhang, C.: Learning Policy Committees for Effective Personalization in MDPs With Diverse Tasks. Proceedings of the 42nd International Conference on Machine Learning (2025)

10. Ghosh, A., Chung, J., Yin, D., Ramchandran, K.: An Efficient Framework for Clustered Federated Learning. Advances in Neural Information Processing Systems (2020)
11. Gong, B., Xing, T., Liu, Z., Xi, W., Chen, X.: Adaptive Client Clustering for Efficient Federated Learning Over Non-IID and Imbalanced Data. IEEE Transactions on Big Data (2022)
12. Karimireddy, S.P., Kale, S., Mohri, M., Reddi, S., Stich, S., Suresh, A.T.: Scaffold: Stochastic Controlled Averaging for Federated Learning. In: International Conference on Machine Learning (2020)
13. Krizhevsky, A., Hinton, G., et al.: Learning Multiple Layers of Features From Tiny Images. Toronto, ON, Canada (2009)
14. Lang, H., Sontag, D., Vijayaraghavan, A.: Theoretical Analysis of Weak-to-Strong Generalization. Advances in Neural Information Processing Systems (2024)
15. Le, Y., Yang, X.: Tiny ImageNet Visual Recognition Challenge. Stanford CS231N: Convolutional Neural Networks for Visual Recognition (2015)
16. Li, D., Wang, J.: FedMD: Heterogenous Federated Learning via Model Distillation. arXiv preprint arXiv:1910.03581 (2019)
17. Li, T., Hu, S., Beirami, A., Smith, V.: Ditto: Fair and Robust Federated Learning Through Personalization. In: International Conference on Machine Learning (2021)
18. Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V.: Federated Optimization in Heterogeneous Networks. Proceedings of Machine Learning and Systems (2020)
19. Lin, T., Kong, L., Stich, S.U., Jaggi, M.: Ensemble Distillation for Robust Model Fusion in Federated Learning. Advances in Neural Information Processing Systems (2020)
20. Liu, J., Liu, X., Wang, S., Wan, X., Li, D., Lu, K., He, K.: Communication-efficient federated multi-view clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence **48**, 17–32 (2025), `https://api.semanticscholar.org/CorpusID:280767120`
21. Makhija, D., Han, X., Ho, N., Ghosh, J.: Architecture Agnostic Federated Learning for Neural Networks. In: International Conference on Machine Learning (2022)
22. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-Efficient Learning of Deep Networks from Decentralized Data. In: International Conference on Artificial Intelligence and Statistics (2017)
23. Mora, A., Tenison, I., Bellavista, P., Rish, I.: Knowledge Distillation in Federated Learning: A Practical Guide. In: Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (2024)
24. Oh, J., Kim, S., Yun, S.Y.: Fedbabu: Toward enhanced representation for federated image classification. In: International Conference on Learning Representations (2022)
25. Sattler, F., Marban, A., Rischke, R., Samek, W.: Cfd: Communication-efficient federated distillation via soft-label quantization and delta coding. In: IEEE Transactions on Network Science and Engineering (2022)
26. Shahid, O., Pouriyeh, S., Parizi, R.M., Sheng, Q.Z., Srivastava, G., Zhao, L.: Communication efficiency in federated learning: Achievements and challenges. arXiv preprint arXiv:2107.10996 (2021)
27. Shamsian, A., Navon, A., Fetaya, E., Chechik, G.: Personalized Federated Learning Using Hypernetworks. In: International Conference on Machine Learning (2021)
28. Tamirisa, R., Xie, C., Bao, W., Zhou, A., Arel, R., Shamsian, A.: FedSelect: Personalized Federated Learning With Customized Selection of Parameters for Fine-

Tuning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2024)
29. Tan, A.Z., Yu, H., Cui, L., Yang, Q.: Towards Personalized Federated Learning. IEEE Transactions on Neural Networks and Learning Systems (2022)
30. Tan, Y., Long, G., Liu, L., Zhou, T., Lu, Q., Jiang, J., Zhang, C.: FedProto: Federated Prototype Learning Across Heterogeneous Clients. In: Proceedings of the AAAI Conference on Artificial Intelligence (2022)
31. Wei, C., Shen, K., Chen, Y., Ma, T.: Theoretical Analysis of Self-Training With Deep Networks on Unlabeled Data. In: International Conference on Learning Representations (2020)
32. Yurochkin, M., Agarwal, M., Ghosh, S., Greenewald, K., Hoang, N., Khazaeni, Y.: Bayesian Nonparametric Federated Learning of Neural Networks. In: International Conference on Machine Learning (2019)
33. Zhang, J., Shi, Y.: A Personalized Federated Learning Method Based on Clustering and Knowledge Distillation. Electronics (2024)
34. Zhu, Z., Hong, J., Zhou, J.: Data-Free Knowledge Distillation for Heterogeneous Federated Learning. In: International Conference on Machine Learning (2021)