On-Demand Federated Learning for Arbitrary Target Class Distributions

Isu Jeong
UNIST
(Ulsan National Institute
of Science and Technology)

Seulki Lee
UNIST
(Ulsan National Institute
of Science and Technology)

Abstract

We introduce On-Demand Federated Learning (On-Demand FL), which enables on-demand federated learning of a deep model for an arbitrary target data distribution of interest by making the best use of the heterogeneity (non-IID-ness) of local client data, unlike existing approaches trying to circumvent the non-IID nature of federated learning. On-Demand FL composes a dataset of the target distribution, which we call the composite dataset, from a selected subset of local clients whose aggregate distribution is expected to emulate the target distribution as a whole. As the composite dataset consists of a precise yet diverse subset of clients reflecting the target distribution, the on-demand model trained with exactly enough selected clients becomes able to improve the model performance on the target distribution compared when trained with off-target and/or unknown distributions while reducing the number of participating clients and federating rounds. We model the target data distribution in terms of class and estimate the class distribution of each local client from the weight gradient of its local model. Our experiment results show that On-Demand FL achieves up to 5% higher classification accuracy on various target distributions just involving 9× fewer clients with FashionMNIST, CIFAR-10, and CIFAR-100.

Proceedings of the 27th International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238. Copyright 2024 by the author(s).

1 INTRODUCTION

Federated learning (FL) [Konečný et al., 2016] enables data in a large quantity to be learnable in a distributed manner with a set of local clients without explicit data movement. However, existing methods, such as FedAvg [McMahan et al., 2017], are known to suffer from unstable learning and poor performance [Sattler et al., 2019, Li et al., 2019], especially with *heterogeneous (non-IID) local data distributions*. Moreover, a single global FL model can hardly generalize to *various data distributions*, e.g., local client-specific datasets [Zhao et al., 2018].

Many approaches, e.g., clustered federated learning (CFL) [Sattler et al., 2020a], federated multi-task learning (FMTL) [Smith et al., 2017], personalized federated learning (PFL) [Hanzely and Richtárik, 2020], etc., have been proposed to achieve effective federated learning on heterogeneous (non-IID) data distributions. Although they can improve the performance of federated learning by mitigating the data heterogeneity to some degree, their scope is still limited to *a pre-defined and fixed data distribution*, e.g., IID, client-specific, or clustered distributions, which does not flexibly apply to *arbitrary and multifarious data distributions.*

Unlike existing works that regard data heterogeneity (non-IID) as an obstacle to overcome [Zhu et al., 2021], we argue that the non-IID-ness is natural and intrinsic in federated learning and can be even beneficial when carefully exploited. By actively leveraging the heterogeneity (non-IID) of local client data instead of trying to surmount it, it becomes possible to provide high-performing models on diverse and ever-evolving non-stationary data distributions. In particular, we consider a number of disparate local client data as the collective pool of a diverse and rich dataset from which we can flexibly derive an arbitrary target distribution of interest by composing an adequate subset of clients.

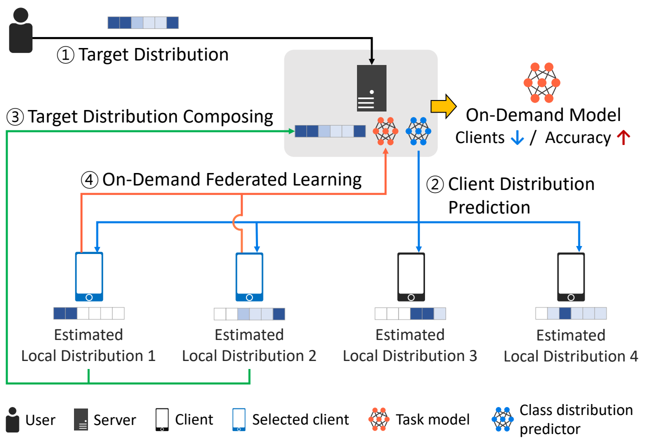
① Target Distribution
③ Target Distribution Composing
On-Demand Model
Clients ↓ / Accuracy ↑
④ On-Demand Federated Learning
② Client Distribution Prediction
Estimated Local Distribution 1
Estimated Local Distribution 2
Estimated Local Distribution 3
Estimated Local Distribution 4
User
Server
Client
Selected client
Task model
Class distribution predictor

Figure 1: Given an arbitrary on-demand target distribution, a subset of local clients is selected to emulate (compose) the target distribution based on the estimated local data distributions. Then, the on-demand model is federated-learned with the selected subset, optimizing it for the target distribution, enhancing performance while involving fewer clients in model training.

In this paper, we propose *On-Demand Federated Learning* (*On-Demand FL*)—a new FL paradigm capable of optimizing a deep classification model to *an arbitrary target distribution* on demand, including physically non-existing distributions, requested from a user, especially given heterogeneous (non-IID) distributions. The specified target distribution is composed of a subset of clients' local datasets, called *the composite dataset*, selected to collectively constitute the target distribution. Given the diverse data distributions among local clients in federated learning, the desired target distribution can be effectively formed by combining local clients appropriately. With the composite dataset, On-Demand FL produces an *on-demand model* customized to the target distribution, while *a much smaller number of local clients* participates in federated learning. Thus, the idea of On-Demand FL can be seen as a generalization of existing works that produce a single global [Zhang et al., 2021], several clustered [Sattler et al., 2020a], or many personalized models [Tan et al., 2022].

We argue that On-Demand FL possesses inherent properties that render it a natural and advantageous choice in many real business and industry scenarios.

Sharing Economy In the sharing economy [David, 2017], resources are shared instead of owned, leading to frequently changing users. Traditional intelligent systems struggle to train personalized models for individual users. However, On-Demand FL allows users to efficiently use a system tailored to each individual's data class distribution by simply providing their usage patterns' data class distribution.

Industrial Internet of Things (IIoT) On-Demand FL is also suitable for IIoT [Boyes et al., 2018], where a multitude of sensors monitors the industrial environment and control processes. Since On-Demand FL does not have to communicate with all clients in the industrial field, it can select only a necessary subset of clients required to construct the target distribution with fewer clients than the entire client pool, enabling efficient model training for the desired target class distribution.

Figure 1 depicts the four steps of On-Demand FL, which are described as follows.

(Fig 1-①) On-Demand FL first takes the desired target distribution of interest from a user or application. When dealing with data distribution, On-Demand FL utilizes the *class distribution* of data, defined as the relative ratio of each class, which is practical, efficient, and well-fit to describe and compose the desired target distribution. From a user's perspective, specifying the desired target distribution in terms of class is intuitive and user-friendly, not entailing complex technicalities.

(Fig 1-②) Before composing the target class distribution from a subset of clients, the local class distribution of each client is estimated. To this end, we propose the *class distribution predictor*, a deep neural network (DNN) that predicts the local data distribution of a client from the weight gradient of the local model, similar to conventional federated learning (e.g. FedAvg [McMahan et al., 2017]) that uses only the weight parameters for model training [Konečný et al., 2016]. The class distribution predictor is trained via regular federated learning, without requiring to modify any conventional federated learning protocols in contrast to previous approaches [Sattler et al., 2020a, Smith et al., 2017]. The run-time estimation of local class distributions enables training an on-demand model for the target class distribution, even if client distributions change due to continuous data acquisition.

(Fig 1-③) Once the local class distribution of each client is estimated, the composite dataset is organized from a selected subset of clients such that their aggregate class distribution becomes close to the target class distribution. The selected clients not only match the target class distribution but also improve the performance of the on-demand model. They maintain the precise target data distribution while collectively increasing the diversity of the composite dataset, thus enhancing the potential learning capability of the on-demand model.

(Fig 1-④) Finally, on-demand federated learning starts to train the on-demand model for the target class distribution only using the selected clients. Starting from the model trained with all clients, the on-demand model is fine-tuned to fit the composite dataset of the target distribution, achieving improved learning performance with fewer local clients being involved.

We evaluate On-Demand FL on FahionMNIST [Xiao et al., 2017], CIFAR-10, and CIFAR-100 [Krizhevsky et al., 2009], achieving improved model performance on various target distributions, while a much less number of clients participate in federated learning, e.g., 5% higher classification accuracy on CIFAR-10 with 9× fewer clients. It shows that On-Demand FL improves the flexibility, training efficiency, and even model performance of federated learning for arbitrary class distributions, extending the capability of federated learning in non-IID setups beyond the pre-defined and fixed data configurations.

We implement the proposed On-Demand FL with PyTorch [Paszke et al., 2017], which is available at a public git repository [1].

2 RELATED WORK

Client Selection Similar to On-Demand FL, some approaches select a subset of participating clients in homogeneous data distributions, trying to improve model generalization capability. FAVOR [Wang et al., 2020] counterbalances the bias introduced by non-IID data and speeds up convergence using deep Q-learning [Mnih et al., 2013]. Another method [Yang et al., 2021] selects a subset of clients based on the combinatorial multi-armed bandit algorithms [Chen et al., 2013] to minimize class imbalance. Although the latter estimates the local class distributions based on the local gradient updates in a similar way to On-Demand FL, they need to be compared against the gradients inferred from an auxiliary dataset transferred to the server, violating the privacy-preserving property of federated learning. On the contrary, On-Demand FL transmits only the weight gradient, not any local data or distributions, to the server, so that it keeps data privacy and efficiency in the same way as the original federated learning [Konečný et al., 2016].

Clustered Federated Learning To improve the generalization performance of the model, clustered federated learning (CFL) has been introduced in which clients are grouped into several clusters with jointly trainable data distributions [Sattler et al., 2020a, Briggs et al., 2020, Sattler et al., 2020b]. Although it improves the accuracy of clustered models by examining the geometric properties of the loss surface, it comes with high computation and communication costs of the recursive bi-partitioning process with similarity measurements between the gradient updates of clients, making it difficult to scale out when the number of clients increases [Ghosh et al., 2020]. In contrast, On-Demand FL conducts large-scale federated learning without repetitive clustering, selecting clients only once.

Personalized Federated Learning Although per-cluster models perform better than a single global model, it is not guaranteed to find optimal clusters and generalize well for all clients in the same cluster. Alternatively, personalized federated learning (PFL) optimizes an independent local model for each individual client, tailored to a particular distribution of local data [Tan et al., 2022, Arivazhagan et al., 2019]. Other methods [Hanzely and Richtárik, 2020] learn personalized models using a mixture of global and local models to balance generalization with personalization. Similarly, APFL [Deng et al., 2020] and others [Mansour et al., 2020] try to find the optimal combination of global and local models in a communication-efficient manner using a weighting factor on a specific local model. FedFomo [Zhang et al., 2020] generates a personalized model for specific data distribution but relies on client's validation datasets, and exchanges local models between clients. On-Demand FL can also be considered as a variant of personalized federated learning as it optimizes an individual model to best fit a particular data distribution. However, On-Demand FL is capable of deriving custom models optimized for any desired target distributions by composing them from diverse local datasets.

Federated Multi-Task Learning By treating each client as a single task in multi-task learning (MTL) [Zhang and Yang, 2018, Caruana, 1997], federated multi-task learning (FMTL) learns a better local model on heterogeneous local data from the relationships across clients. MOCHA [Smith et al., 2017] uses a primal-dual formulation to optimize a personalized model for each client by extending distributed MTL into federated learning. However, it is not suitable for many real-world applications as all clients are required to participate in every round of model training. Also, it does not apply to deep neural networks as it is devised only for convex models. Although VIRTUAL [Corinzia et al., 2019] can handle non-convex models by using a variational inference, it is computationally expensive for large-scale cases. FedAMP [Huang et al., 2021] is an attention-based mechanism that encourages collaboration among clients with similar distributions in which a personalized model becomes the linear combination of local client models. However, its storage and communication overhead increase linearly as the client's personalized model is maintained on the server. Unlike such methods, On-Demand FL does not modify existing federated learning mechanisms and communication protocols, making it easy to implement in practice.

[1] `https://github.com/eai-lab/On-DemandFL`

3 CLIENT DISTRIBUTION PREDICTION

We first describe the concept of class distribution used in On-Demand FL and the class distribution predictor that estimates the class distribution of a client from the weight gradient of its local model.

3.1 Class Distribution

Class Distribution For the client device c having a total of n classes, its class distribution vector $\vec{d_c}$ is composed of n distribution factors, d_c^i for $1 \leq i \leq n$, where d_c^i represents the relative ratio of data samples belonging to the i-th class in the local dataset available on the client c as:

$$\vec{d_c} = [d_c^1 \; d_c^2 \; \cdots \; d_c^n] \quad \text{where} \quad d_c^i = \frac{|D_c^i|}{\sum_{k=1}^{n} |D_c^k|} \tag{1}$$

Here, D_c is the local dataset on the client c, D_c^i is the set of data samples of the i-th class, and $|D_c^i|$ is the number of samples in D_c^i. The sum of class distribution vector $\vec{d_c}$ is to be one, i.e., $\sum_{i=1}^{n} d_c^i = 1$.

Privacy-Preserving Prediction Following the spirit of federated learning [McMahan et al., 2017], which trains a model on local clients and only transmits the local weight parameters without data movement for privacy preservation and communication efficiency, On-Demand FL estimates the class distribution vector $\vec{d_c}$ of the client c without directly exposing data or class distribution outside of the client. In particular, the class distribution is estimated from only the weight gradient of the local model without exchanging any other data between the client and server, minimizing the risk of potential data leakages.

From Gradient to Class Distribution As briefly mentioned above, On-Demand FL takes the weight gradient of a local client model as the input for class distribution prediction due to its effectiveness and efficiency in finding class distribution patterns. We assume that the primary task of On-Demand FL is a classification deep neural network (DNN) having a softmax output at the last layer with the cross-entropy loss L_t, which is given by:

$$L_t(x^j) = -\sum_{i=1}^{k} \tilde{y}_i^j \log y_i^j \quad \text{where } y^j = \text{Softmax}(h^{l,j}) \quad \text{and } h^{l,j} = h^{l-1,j} w^l + b^l \tag{2}$$

Here, $\{x^j, \tilde{y}^j\}$ is the input-output pair of the j-th data sample, i.e., $\tilde{y}_i^j \in \{0,1\}$ is the ground-truth label of the input x^j on the i-th class, y_i^j is the softmax output of the i-th class on x^j, w^l and b^l is the weight parameter and bias set of the last layer l, respectively, and $h^{l,j}$ and $h^{l-1,j}$ is the output at the last and penultimate layer given x^j, respectively.

Given Equation 2, the gradient of $L_t(x^j)$ w.r.t. the weight parameter w^l is obtained by the chain rule.

$$\begin{aligned}
\frac{\partial L_t(x^j)}{\partial w^l} &= \frac{\partial L_t(x^j)}{\partial y^j} \frac{\partial y^j}{\partial h^{l,j}} \frac{\partial h^{l,j}}{\partial w^l} \\
&= \left[\hat{y}_1^j \; \hat{y}_2^j \cdots \hat{y}_n^j\right] \begin{bmatrix} y_1^j(1-y_1^j) & \cdots & -y_1^j y_n^j \\ -y_2^j y_1^j & \cdots & -y_2^j y_n^j \\ \vdots & \vdots & \vdots \\ -y_n^j y_1^j & \cdots & y_n^j(1-y_n^j) \end{bmatrix} \frac{\partial h^{l,j}}{\partial w^l} \\
&= \left[y_1^j - \tilde{y}_1^j \; y_2^j - \tilde{y}_2^j \cdots y_n^j - \tilde{y}_n^j\right] \frac{\partial h^{l,j}}{\partial w^l}
\end{aligned} \tag{3}$$

where $\hat{y}_i^j = -1/y_i^j$ if $\tilde{y}_i^j = 1$, and $\hat{y}_i^j = 0$ otherwise. Since the ground-truth label $\tilde{y}^j$ is a one-hot vector, only one of them is non-zero, i.e., $\hat{y}_i^j = -1/y_i^j$ for i and $\hat{y}_k^j = 0$ for all $k \neq i$ when $\tilde{y}_i^j = 1$. For example, when $\tilde{y}_1^j = 1$ and $\tilde{y}_i^j = 0$ for all $i \neq 1$, i.e., $\tilde{y}^j = [1 \; 0 \cdots 0]$, the first component of Equation 3 becomes $\partial L_t(x^j)/\partial y^j = [-1/y_1^j \; 0 \cdots 0]$, and the dot product of the first and second components gives $\partial L_t(x^j)/\partial y^j \cdot \partial y^j/\partial a^{l,j} = [y_1^j - 1 \; y_2^j \cdots y_n^j]$, which results in the following.

$$\frac{\partial L_t(x^j)}{\partial w^l} = \left[y_1^j - 1 \; y_2^j \cdots y_n^j\right] \frac{\partial h^{l,j}}{\partial w^l} \quad \text{if } \tilde{y}_1^j = 1 \tag{4}$$

where $-1 \leq y_1^j - 1 \leq 0$, and $0 \leq y_i^j \leq 1$ for all $i \neq 1$ since y^j is softmax, i.e., $0 \leq y_i^j \leq 1$ for all i.

Given $\partial h^{l,j}/\partial w^l = h^{l-1,j}$ is at least zero as $h^{l,j}$ and $h^{l-1,j}$ are the outputs of the last layer and penultimate layer, respectively, the first column of the matrix $\partial L_t(x^j)/\partial w^l$ in Equation 4 becomes negative, while the other columns remain non-negative. Hence, the summation of the gradient (Equation 3) over k data samples (mini-batch) of the client c can be represented by the class distribution factor d_c^i in Equation 1 as:

$$\begin{aligned}
\sum_{j=1}^{k} \frac{\partial L_t(x^j)}{\partial w_c^l} &= \sum_{j=1}^{k} \left[y_1^j - \tilde{y}_1^j \cdots y_n^j - \tilde{y}_n^j\right] \frac{\partial h^{l,j}}{\partial w_c^l} \\
&\propto \sum_{i=1}^{n} d_c^i \left[s_1 \cdots s_n\right] \sum_{j \in \text{class } i}^{k} \frac{\partial h^{l,j}}{\partial w_c^l}
\end{aligned} \tag{5}$$

where s_b is the sign indicator on the b-th column, i.e., $-1 \leq s_b \leq 0$ if $b = i$ and $0 \leq s_b \leq 1$ otherwise for all $1 \leq b \leq n$. By assuming that $\lim_{k\to\infty} \sum_{j \in \text{class } i}^{k} \partial h^{l,j}/\partial w_c^l$ converges to a constant, we hypothesize that the gradient summation

$\sum_{j=1}^{k} \partial L_t(x^j)/\partial w_c^l$ reflects the class distribution pattern of the i-th class, d_c^i, as $\sum_{j=1}^{k} \partial L_t(x^j)/\partial w_c^l \propto \sum_{i=1}^{n} d_c^i [s_1\ s_2 \cdots s_n]$. As a result, we can make reliable inferences on the i-th class distribution factor d_c^i of the client c from the weight gradient sum $\sum_{j=1}^{k} \partial L_t(x^j)/\partial w_c^l$ obtained from the local client model.

3.2 Class Distribution Predictor

Class Distribution Prediction Based on the class distribution analysis given above, we propose a class distribution predictor DNN (deep neural network) that provides the estimated class distribution of local client data from the weight gradient of the local client model.

Given the weight parameter set at the last layer of the local task model, w_c^l, on the client c, the class distribution predictor C_p estimates the class distribution vector, $\vec{d_c}$, defined in Equation 1 as follows:

$$\sum_{x^j \in D_c} C_p\left(g^{w_c^l}(x^j)\right) = \vec{d_c} \text{ where } g^{w_c^l}(x^j) = \frac{\partial L_t(x^j)}{\partial w_c^l} \quad (6)$$

Here, $g^{w_c^l}(x^j)$ is the gradient of the weight parameter set w_c^l given x^j on the client c in Equation 3, and D_c is the entire dataset of the client c. We consider that $\sum_{j=1}^{k} g^{w_c^l}(x^j)$ (Equation 5) with a reasonable size of k or a mini-batch of $g^{w_c^l}(x^j)$ can approximate the gradient over the entire dataset $\sum_{x^j \in D_c} g^{w_c^l}(x^j)$, and thus feed $\sum_{j=1}^{k} g^{w_c^l}(x^j)$ to the class distribution predictor C_p. The class distribution predictor DNN consists of n fully-connected layers, as the amount of the weight gradient at the last layer of local models is relatively small, enabling efficient training of the class distribution predictor locally on clients with affordable overhead.

Training of Class Distribution Predictor The class distribution predictor is trained through regular federated learning, e.g., FedAvg [McMahan et al., 2017], in parallel with the task model. For each round of federating iteration, the task model is trained with a batch of k local data samples in client update, executing the local update of its weight parameters, including the last layer w_c^l. After several local training (federating) iterations of the task model, a pair of training examples for the class distribution predictor is produced as $\{g^{w_c^l}, \vec{d_c^k}\}$ where $\vec{d_c^k}$ is the class distribution vector of k data samples of client c.

Once several pairs of training examples, $\{g^{w_c^l}, \vec{d_c^k}\}$, are collected as a mini-batch, the class distribution predictor is trained with them by executing the local update of its own weight parameters. Specifically, it is trained to minimize the mean squared error (MSE). When the local client update is complete, the server aggregates the weight parameters of each local client to generate the global class distribution predictor via federated learning. As the training procedure of the class distribution predictor follows the regular federated learning protocol, any existing weight-based techniques can be used for weight aggregation, such as FedAvg [McMahan et al., 2017], Scaffold [Karimireddy et al., 2019], FedMD [Li and Wang, 2019], and FedDF [Lin et al., 2020].

4 ON-DEMAND FEDERATED LEARNING

Once the class distribution of each client is estimated through the class distribution predictor DNN, a subset of clients is selected to organize the composite dataset whose aggregate distribution emulates the target class distribution. Then, the on-demand model is optimized (trained) for the target class distribution by performing on-demand federated learning with the selected clients.

4.1 Target Distribution Composing

Given the target class distribution vector for n classes as $\vec{d_t} = [d_t^1\ d_t^2 \cdots d_t^n]$ and the estimated class distribution vector for the k-th client as $\vec{d_{c_k}} = [d_{c_k}^1\ d_{c_k}^2 \cdots d_{c_k}^n]$, a subset of clients is chosen from a total of m clients to perform On-Demand FL by solving the linear equation.

$$\begin{bmatrix} d_t^1 \\ \vdots \\ d_t^n \end{bmatrix} = \begin{bmatrix} d_{c_1}^1 & d_{c_2}^1 & \cdots & d_{c_m}^1 \\ \vdots & \vdots & \ddots & \vdots \\ d_{c_1}^n & d_{c_2}^n & \cdots & d_{c_m}^n \end{bmatrix} \begin{bmatrix} r_{c_1} \\ \vdots \\ r_{c_m} \end{bmatrix} \quad (7)$$

The ratio vector $\vec{r} = [r_{c_1}\ r_{c_2} \cdots r_{c_m}]$ ($\vec{r} \geq 0$) on the right-hand side denotes the relative ratio of each client we want to find whose summation is one, i.e., $\sum_k r_{c_k} = 1$. In order to determine $\vec{r}$, we solve Equation 7 by using Non-Negative Least Squares (NNLS) [Lawson and Hanson, 1995] that finds the best $\hat{x}$ when it is supposed to have no solution x in the $Ax = b$ ($x \geq 0$) problem. Applying NNLS to Equation 7 gives:

$$\arg\min_{\vec{r}} ||\mathcal{D} \cdot \vec{r} - \vec{d_t}||_2 \quad \text{s.t. } \vec{r} \geq 0 \quad (8)$$

where $\mathcal{D}$ denotes the estimated distribution matrix having $\vec{d_{c_k}}$ as column vectors for clients, i.e., the first component on the right-hand side of Equation 7.

Once having the ratio vector $\vec{r}$, the clients with non-zero ratio value, $r_{c_i} > 0$, are selected as the subset S_s of the entire client set S_m as follows.

$$S_s = \{c_i | c_i \in S_m \wedge r_{c_i} > 0\} \quad (9)$$

Then, in every round of on-demand federated learning, a fraction of the selected clients, denoted as $S_r \subseteq S_s$, is randomly sampled for the model update by using $\vec{r}$ as the client distribution of S_r as:

$$S_r = \{c_i | c_i \in S_s\} \text{ s.t. } S_r \sim \vec{r} \text{ and } |S_r| = |S_s| \times f \quad (10)$$

where f is the federated learning fraction. Here, $\vec{r}$ is used as the per-client ratio, working as a client selection factor that maintains the target class distribution.

4.2 On-Demand FL Algorithm

Algorithm 1 describes the entire procedure of On-Demand FL. Each client is indexed by c, S_m is the entire set of m clients, w_t and w_p are the weight parameters of the task model and the class distribution predictor, respectively, η_t and η_p is the learning rate of the task model and the class distribution predictor, respectively, and n_t and n_p is the number of data samples and weight gradients, respectively, used for a single round of federated learning. In short, On-Demand FL first 1) trains the task model and class distribution predictor with federated learning manner, then 2) selects a subset of clients to organize the composite dataset of the target class distribution based on their estimated class distributions, and finally 3) optimizes the on-demand model to the target class distribution with selected clients. The backbone federated learning algorithm that updates the task model and the class distribution predictor is compatible with various federated learning algorithms, e.g., regularized FL [Li et al., 2020, Karimireddy et al., 2019, Acar et al., 2021]. In this paper, we use FedAvg [McMahan et al., 2017] as the backbone federated learning algorithm.

Algorithm 1 On-Demand Federated Learning

```
Input: Client set $S_m$, local dataset $D_c$ on each client $c \in S_m$, and target class distribution vector $\vec{d_t}$
Output: On-demand model $w_t$ optimized for $\vec{d_t}$
On-Demand FL begin
    FedLearning($S_m, w_t, w_p$)                                    [train of $C_p$]
    $S_s \leftarrow$ select clients for $\vec{d_t}$ from $S_m$      [Eq 8 and 9]
    FedLearning($S_s, w_t, \emptyset$)                              [train of $w_t$]
end
FedLearning($S, w_t, w_p$) begin
    for each round $r = 1, 2, 3, \cdots,$ do
        $S_r \leftarrow$ random subset of $S$   [using Eq 10 if $w_p = \emptyset$]
        for each client $c \in S_r$ do
            if $w_p = \emptyset$ then $(w_t^c, \emptyset) \leftarrow$ ClientUpdate($w_t^c, \emptyset$)
            else $(w_t^c, w_p^c) \leftarrow$ ClientUpdate($w_t^c, w_p^c$)
        end
        if $w_p \neq \emptyset$ then $w_t \leftarrow \sum_c \frac{n_t^c}{n_t} w_t^c$ and $w_p \leftarrow \sum_c \frac{n_p^c}{n_p} w_p^c$
        else $w_t \leftarrow \sum_c r^c \frac{n_t^c}{n_t} w_t^c$ [$r_c$ calculated in Eq 7 and 8]
    end
end
ClientUpdate($w_t^c, w_p^c$) begin
    $G \leftarrow \emptyset$
    for each local epoch $i_t \in \{1, ..., E_t\}$ do
        for $x^j \in$ random batch of $D_c$ (dataset of client c)
         do
            $w_t^c \leftarrow w_t^c - \eta_t \frac{\partial L_t(x^j)}{\partial w_t^c}$
            if $w_p^c \neq \emptyset$ then $G \leftarrow G \cup \frac{\partial L_t(x^j)}{\partial w_{t,c}^l}$ ;
        end
    end
    for each local epoch $i_p \in \{1, ..., E_p\}$ do
        if $w_p^c \neq \emptyset$ then for each gradient $g^{w_{t,c}^l} \in G$ do
            $w_p^c \leftarrow w_p^c - \eta_p \frac{\partial L_p(g^{w_{t,c}^l})}{\partial w_p^c}$          [$C_p$ update]
        end
    end
    return $(w_t^c, w_p^c)$ to server
end
```

5 EXPERIMENTS

Implementation We implement On-Demand FL with PyTorch [Paszke et al., 2019] and open-source it with an anonymous public git repository [1].

Experimental Setting Table 1 summarizes three experimental settings we use to evaluate the proposed On-Demand FL. The details of the experimental settings are given as follows.

Table 1: Three experimental settings with different datasets, class numbers, and task models.

Experiment Setting	Dataset	Class Num	Task Model	Class Distribution Predictor
1	FashionMNIST	10	LEAF (2 conv + 2 fc)	3 layer fc
2	CIFAR-10	10	LEAF (2 conv + 2 fc)	
3	CIFAR-100	100	MobileNetV2	

Dataset and Client Distribution We conduct experiments on three datasets, i.e., FahionMNIST, CIFAR-10, and CIFAR-100 [Xiao et al., 2017, Krizhevsky et al., 2009]. Their classes are allocated to each local client based on the Dirichlet distribution [Yurochkin et al., 2019]. We set the distribution of the i-th class on m clients (i.e., from c_1 to c_m) as $\vec{p^i} = [p_{c_1}^i, p_{c_2}^i, \cdots, p_{c_m}^i] \sim Dirichlet(\alpha)$ with parameter $\alpha \in \{0.1, 0.5, 1.0\}$ such that $\sum_{k=1}^{m} p_{c_k}^i = 1$. The parameter α determines the degree of IID-ness of data; a smaller α generates a more non-IID distribution and vice versa. Figure 2 shows examples of client distribution on CIFAR-100.

Target Distribution (TD) Two types of target distributions (TD) are constructed for the experiment. First, we construct TD classified into three groups according to the ratio of classes of interest (TD 1, 2, and 3), such that each TD only contains data samples belonging to the class $\{c_1, \cdots, c_k\}$, where $k \in \{3 \times N, 4 \times N, 5 \times N\}$ and N is the number of classes divided by 10. Second, we construct a set of TD mapped with the local class distribution of clients (Local). Figure 3 depicts a visual representation of the four TDs.

Task Models For FashionMNIST and CIFAR-10, we use a variant of the CNN model, which consists of two convolution and two fully-connected layers, proposed by LEAF [Caldas et al., 2018], a popularly used benchmark framework for federated learning. For CIFAR-100, we use MobileNetV2 [Sandler et al., 2018].

Class Distribution Predictor We build the class distribution predictor DNN as a fully-connected network with variations in the architecture depending on the dataset, task model, and other factors.

5.1 Performance of On-Demand Model

As the first evaluation, the end-to-end performances of On-Demand FL and On-Demand FL (GT) are assessed by comparing the classification accuracy of on-demand models on the four target distributions against those obtained by four baseline methods, i.e., FedAvg [McMahan et al., 2017], clustered federated learning (CFL) [Sattler et al., 2020a], federated multi-task learning (FMTL), MOCHA [Smith et al., 2017] in particular, and personalized federated learning (FedFOMO) [Zhang et al., 2020]. On-Demand FL (GT) selects client subsets based on their ground truth class distribution. We generate a total of m=100 clients from the Dirichlet distribution with $\alpha \in \{0.1, 0.5, 1.0\}$ and the federated learning fraction f=0.1 and test them with the four target distributions, i.e., TD 1, 2, 3, and Local client distributions. We allow On-Demand FL to select at least $m \times f$=10 clients as a subset by repeatedly applying NNLS (Non-Negative Least Squares) [Lawson and Hanson, 1995] to ensure a minimum number of clients and data required by federated learning when composing the target distributions (TDs).

Table 2 summarizes the result. Overall, On-Demand FL outperforms or performs at least competitively to all the baselines for TD 1, 2, 3, and Local TDs regardless of the non-IID-ness represented by the parameter α, as it fine-tunes the on-demand model with a diverse subset of clients that collectively reflects the target distribution. The slight performance difference between On-Demand FL and On-Demand FL (GT) indicates

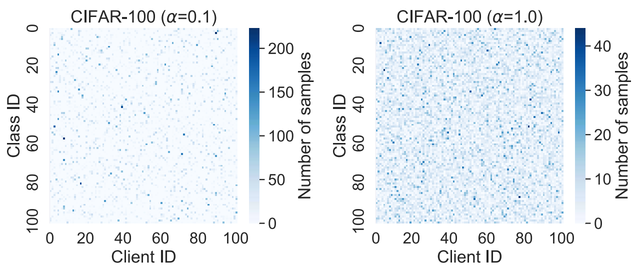

Figure 2: The client distributions generated from the Dirichlet distribution with different α for the CIFAR-100 dataset on 100 clients.

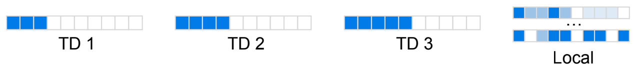

Figure 3: The four target distributions (TD). Each cell represents a class, with color indicating sample count; the darker, the more data samples.

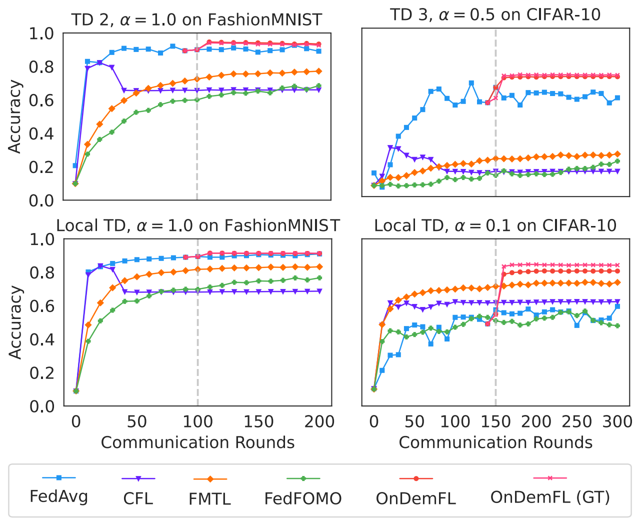

Figure 4: The test classification accuracy of task models over FL rounds optimized by On-Demand FL and the baseline methods. The gray dotted line indicates the starting round of On-Demand FL.

a well-optimized distribution predictor is capable of accurately predicting the class distribution needed to construct a subset. On-Demand FL achieves higher accuracy on FashionMNIST (+0.6%), CIFAR-10 (+5.1%), and CIFAR-100 (+0.9%) on average, compared to the best-performing baseline, respectively, with a subset of clients that are 9.1×, 8.9×, and 5.2× fewer than the entire client pool. As intended, On-Demand FL leverages heterogeneity and outperforms other FL methods in a non-IID environment (α=0.1). Furthermore, On-Demand FL shows competitive or superior performance in a more-IID environment (α=1.0). The results demonstrate that On-Demand FL enables robust and stable federated learning across various TDs with fewer clients, which is not achievable by the baseline methods.

Figure 4 shows the test accuracy of On-Demand FL, FedAvg, CFL, FMTL, and FedFOMO over the federating round on FashionMNIST and CIFAR-10. For TD1, TD3, and Local TDs. On-Demand FL achieves higher accuracy than all the other baseline federated learning methods with large margins (i.e., 7.0% higher accuracy on CIFAR-10 for $\alpha = 0.1$ and local TDs). It notes that On-Demand FL achieves superior or comparable model performance using 8.9× fewer clients on average than the baselines through training over a small number of rounds. That can be further improved by making more clients participate in On-Demand FL.

Table 2: The test classification accuracy of task models obtained by FedAvg, CFL, FMTL, FedFOMO, On-Demand FL, and On-Demand FL (GT) on all three experimental settings. On-Demand FL starts from 100, 150, and 1200 rounds from FedAvg for FashionMNIST, CIFAR-10, and CIFAR-100, respectively. On-Demand FL outperforms in heterogeneous (non-IID) data distributions (i.e., $\alpha = 0.1$ and 0.5) while providing comparable performance in more homogeneous (IID) data distributions (i.e., $\alpha = 1.0$) using up to 9.1× fewer clients.

Experimental Setting	**FL Algorithm**	**Accuracy**											
		$\alpha = 0.1$				$\alpha = 0.5$				$\alpha = 1.0$			
		TD 1	TD 2	TD 3	Local	TD 1	TD 2	TD 3	Local	TD 1	TD 2	TD 3	Local
1 LEAF (FashionMNIST)	FedAvg	0.96	0.95	**0.92**	0.89	0.95	**0.95**	**0.92**	0.92	0.94	0.94	**0.92**	0.91
	CFL	0.72	0.52	0.50	0.87	0.64	0.82	0.84	0.87	0.78	0.82	0.63	0.82
	FMTL	0.33	0.27	0.31	0.92	0.54	0.68	0.71	0.87	0.73	0.77	0.61	0.82
	FedFOMO	0.29	0.27	0.22	0.78	0.46	0.61	0.62	0.81	0.68	0.70	0.54	0.77
	OnDemFL	**0.97**	0.95	0.91	0.95	**0.96**	0.94	**0.92**	**0.93**	0.95	**0.95**	**0.92**	**0.92**
	OnDemFL (GT)	**0.97**	**0.96**	**0.92**	**0.96**	**0.96**	**0.95**	**0.92**	**0.93**	**0.96**	**0.95**	**0.92**	0.91
2 LEAF (CIFAR-10)	FedAvg	0.74	0.71	0.65	0.60	0.75	0.70	0.67	0.68	0.75	0.70	0.66	0.67
	CFL	0.34	0.34	0.34	0.63	0.23	0.31	0.56	0.49	0.29	0.35	0.25	0.42
	FMTL	0.33	0.19	0.19	0.74	0.25	0.28	0.28	0.53	0.29	0.31	0.29	0.47
	FedFOMO	0.33	0.12	0.12	0.58	0.19	0.24	0.25	0.42	0.22	0.26	0.23	0.39
	OnDemFL	**0.82**	0.75	0.69	0.81	0.80	0.75	0.71	0.74	**0.79**	0.75	0.70	**0.72**
	OnDemFL (GT)	**0.82**	**0.78**	**0.71**	**0.85**	**0.81**	**0.76**	**0.73**	**0.75**	**0.79**	**0.76**	**0.71**	**0.72**
3 MobileNetV2 (CIFAR-100)	FedAvg	0.44	0.42	0.42	0.43	0.48	0.46	0.46	0.50	0.47	0.46	**0.46**	**0.51**
	CFL	0.05	0.05	0.06	0.41	0.07	0.07	0.07	0.24	0.08	0.08	0.08	0.19
	FMTL	0.04	0.04	0.04	0.36	0.05	0.05	0.05	0.16	0.05	0.05	0.05	0.12
	FedFOMO	0.01	0.01	0.01	0.19	0.01	0.01	0.01	0.08	0.01	0.01	0.01	0.05
	OnDemFL	0.45	0.43	0.43	0.56	**0.48**	**0.47**	**0.46**	**0.50**	0.47	0.45	0.45	0.47
	OnDemFL (GT)	**0.47**	**0.44**	**0.44**	**0.60**	**0.48**	**0.47**	**0.46**	**0.50**	**0.48**	**0.46**	0.45	0.48

5.2 Class Distribution Prediction

In the second experiment, we evaluate the class distribution predictor. After generating a set of 10 clients with different class distributions from the Dirichlet distribution with $\alpha \in \{0.1, 0.5, 1.0\}$, we measure the difference between the ground-truth class distribution of clients and the predicted class distribution estimated by the class distribution predictor. Table 3 shows the differences between them measured by the mean absolute error (MAE) multiplied by the number of classes. The proposed class distribution predictor consistently achieves lower MAEs (an average of 0.45) in all experiments, indicating it provides accurate estimations.

Table 3: The MAE of the class distribution predictor on FashionMNIST, CIFAR10, and CIFAR-100.

Experimental Setting	**MAE**		
	α=0.1	α=0.5	α=1.0
1 FashionMNIST	1.32	0.81	0.57
2 CIFAR-10	0.84	0.44	0.30
3 CIFAR-100	0.40	0.37	0.40

Figure 5 visualizes the comparison between MAE of the estimated class distribution before and after training of the class distribution predictor. After training, the color representing overall MAE becomes brighter, indicating improved accuracy in estimating class distributions. Although it does not perfectly predict the class distribution of individual client/class, it well reflects the tendency of distribution. We empirically found that it is fairly sufficient to compose a variety of target class distributions with small errors as a whole, as shown in both the previous and next subsections.

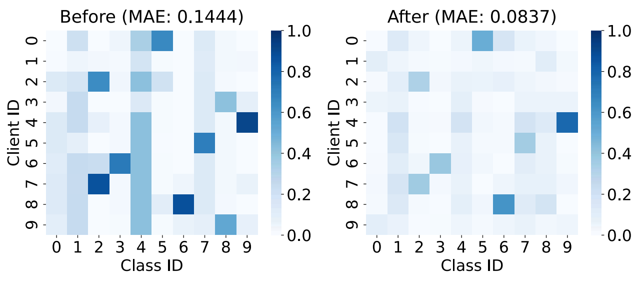

Figure 5: The visualization (heatmap) of the MAE for each client and class before and after training of the class distribution predictor on CIFAR-10 with α=0.1. The lighter, the lower MAE.

5.3 Target Distribution Composing

In the last experiment, we evaluate the difference between the target class distributions and the final class distribution composed by the selected clients. Table 5 summarizes the MAE between them multiplied by the number of data classes. The composed distributions achieve the average MAE of 0.93 in TD 1, 2, 3, and Local TD to target class distributions, showing the effectiveness of the proposed client selection method, which results in high-performance on-demand models while reducing the number of clients involved in FL (i.e., 7.2× fewer clients than entire client pool).

Table 4: The number of rounds needed to reach three accuracy point (AP), i.e., 95%, 98%, and peak point.

AP	The Number of Rounds for OnDemFL								
	FasionMNIST			CIFAR-10			CIFAR-100		
	$\alpha = 0.1$	$\alpha = 0.5$	$\alpha = 1.0$	$\alpha = 0.1$	$\alpha = 0.5$	$\alpha = 1.0$	$\alpha = 0.1$	$\alpha = 0.5$	$\alpha = 1.0$
95%	1.50 ± 2.59	0.43 ± 0.51	0.15 ± 0.35	8.54 ± 12.02	1.13 ± 0.46	0.98 ± 0.5	5.67 ± 3.5	6.67 ± 11.4	26.67 ± 47.15
98%	3.94 ± 5.48	1.02 ± 0.92	0.80 ± 0.47	18.96 ± 22.11	2.59 ± 2.21	2.65 ± 1.73	38.00 ± 37.52	17.17 ± 10.11	76.83 ± 81.9
peak	40.95 ± 32.27	18.45 ± 22.73	18.50 ± 22.89	62.11 ± 51.06	28.66 ± 36.17	29.42 ± 37.33	112.67 ± 81.04	123.50 ± 85.75	130.83 ± 102.01

AP	The Number of Rounds for Backbone Algorithm (FedAvg)								
	FasionMNIST			CIFAR-10			CIFAR-100		
	$\alpha = 0.1$	$\alpha = 0.5$	$\alpha = 1.0$	$\alpha = 0.1$	$\alpha = 0.5$	$\alpha = 1.0$	$\alpha = 0.1$	$\alpha = 0.5$	$\alpha = 1.0$
95%	4.26 ± 7.12	2.37 ± 5.19	0.19 ± 0.8	31.25 ± 38.13	5.48 ± 8.95	2.55 ± 5.46	58.50 ± 56.89	56.33 ± 77.95	45.50 ± 77.79
98%	17.45 ± 19.77	14.92 ± 17.44	5.68 ± 6.78	53.24 ± 47.32	26.04 ± 27.6	18.57 ± 16.34	148.17 ± 69.74	62.50 ± 73.63	45.67 ± 78.16
peak	59.24 ± 29.8	69.61 ± 26.84	64.25 ± 31.09	65.46 ± 47.47	67.76 ± 44.31	54.08 ± 37.37	178.17 ± 76.64	181.83 ± 86.82	94.83 ± 108.05

Table 5: The MAE between the target and the composed distributions organized by the selected clients multiplied by the number of classes. The number in parentheses indicates the number of selected clients among a total of 100 clients.

Experimental Setting	α	MAE			
		TD1	TD2	TD3	Local
1	0.1	1.03 (10)	0.60 (12)	0.98 (10)	0.99 (10.95)
LEAF	0.5	0.63 (12)	0.68 (12)	0.96 (11)	0.82 (11.28)
(FashionMNIST)	1.0	0.64 (10)	0.65 (11)	0.38 (11)	0.62 (11.45)
2	0.1	0.68 (12)	0.87 (11)	0.60 (11)	0.92 (11.32)
LEAF	0.5	1.24 (10)	1.04 (10)	0.83 (11)	0.77 (11.46)
(CIFAR-10)	1.0	1.29 (12)	1.09 (11)	0.88 (13)	0.62 (11.63)
3	0.1	1.03 (21)	0.93 (29)	0.74 (28)	0.95 (18.14)
MobileNetV2	0.5	1.15 (13)	0.95 (14)	0.82 (20)	0.78 (20.16)
(CIFAR-100)	1.0	1.23 (13)	1.04 (15)	0.87 (20)	0.64 (19.84)

5.4 Overhead of On-Demand FL

On-Demand FL requires additional on-demand model training when requested a new target distribution. To measure the overhead of On-Demand FL, we count the number of federated learning rounds as the proxy of the computational complexity required to train the on-demand model under a target distribution. Table 4 presents the number of rounds for On-Demand FL and our backbone algorithm (FedAvg) to reach three accuracy points (AP), i.e., 95%, 98%, and the peak point from the point of On-Demand FL initialization. The results demonstrate that On-Demand FL outperforms FedAvg in terms of efficiency, achieving a reduction of 4.0×, 2.4×, and 1.5 × in the number of rounds needed to reach each AP compared to FedAvg.

6 LIMITATIONS

There are some limitations that can be addressed in future work. First, On-Demand FL necessitates a class distribution predictor. We view this as a reasonable trade-off, offering advantages such as smaller client subsets, swift convergence, and the ability to generate task models for different target distributions. While receiving class distribution directly from clients is more efficient, we believe predicting it from the model parameters is a natural and secure method against potential sniffing. Second, although On-Demand FL requires additional on-demand model training when requesting a new target distribution, it demonstrates rapid accuracy improvement, as shown in Table 4. Moreover, it reduces communication costs compared to other federated learning methods with a small client pool. Our future research aims to minimize training costs by concurrently training models for various target distributions and grouping similar ones.

7 CONCLUSION

We introduce On-Demand FL, a new paradigm of federated learning which optimizes a deep model for an arbitrary target class distribution. To estimate the class distribution of clients, we propose the class distribution predictor that takes the weight gradient of the local model on a client. Once the composite dataset is organized from a set of selected clients, the on-demand model is trained via federated learning. The evaluation shows On-Demand FL achieves up to 5% higher accuracy on the target distribution using 9× fewer clients compared to SOTA federated learning methods.

Acknowledgments

This work was supported by Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT)(No.2020-0-01336, Artificial Intelligence graduate school support(UNIST)) and the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (RS-2023-00277383).

References

[Acar et al., 2021] Acar, D. A. E., Zhao, Y., Navarro, R. M., Mattina, M., Whatmough, P. N., and Saligrama, V. (2021). Federated learning based on dynamic regularization. *arXiv preprint arXiv:2111.04263*.

[Arivazhagan et al., 2019] Arivazhagan, M. G., Aggarwal, V., Singhan, A. K., and Choudhary, S. (2019). Federated learning with personalization layers. *arXiv preprint arXiv:1912.00818*.

[Boyes et al., 2018] Boyes, H., Hallaq, B., Cunningham, J., and Watson, T. (2018). The industrial internet of things (iiot): An analysis framework. *Computers in Industry*, 101:1–12.

[Briggs et al., 2020] Briggs, C., Fan, Z., and Andras, P. (2020). Federated learning with hierarchical clustering of local updates to improve training on non-iid data. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE.

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

[Caruana, 1997] Caruana, R. (1997). Multitask learning. *Machine learning*, 28(1):41–75.

[Chen et al., 2013] Chen, W., Wang, Y., and Yuan, Y. (2013). Combinatorial multi-armed bandit: General framework and applications. In *International conference on machine learning*, pages 151–159. PMLR.

[Corinzia et al., 2019] Corinzia, L., Beuret, A., and Buhmann, J. M. (2019). Variational federated multi-task learning. *arXiv preprint arXiv:1906.06268*.

[David, 2017] David, M. (2017). Sharing: post-scarcity beyond capitalism? *Cambridge Journal of Regions, Economy and Society*, 10(2):311–325.

[Deng et al., 2020] Deng, Y., Kamani, M. M., and Mahdavi, M. (2020). Adaptive personalized federated learning. *arXiv preprint arXiv:2003.13461*.

[Ghosh et al., 2020] Ghosh, A., Chung, J., Yin, D., and Ramchandran, K. (2020). An efficient framework for clustered federated learning. *Advances in Neural Information Processing Systems*, 33:19586–19597.

[Hanzely and Richtárik, 2020] Hanzely, F. and Richtárik, P. (2020). Federated learning of a mixture of global and local models. *arXiv preprint arXiv:2002.05516*.

[Huang et al., 2021] Huang, Y., Chu, L., Zhou, Z., Wang, L., Liu, J., Pei, J., and Zhang, Y. (2021). Personalized cross-silo federated learning on non-iid data. In *AAAI*, pages 7865–7873.

[Karimireddy et al., 2019] Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S. J., Stich, S. U., and Suresh, A. T. (2019). Scaffold: Stochastic controlled averaging for on-device federated learning.

[Konečnỳ et al., 2016] Konečnỳ, J., McMahan, H. B., Ramage, D., and Richtárik, P. (2016). Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*.

[Krizhevsky et al., 2009] Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

[Lawson and Hanson, 1995] Lawson, C. L. and Hanson, R. J. (1995). *Solving least squares problems*. SIAM.

[Li and Wang, 2019] Li, D. and Wang, J. (2019). Fedmd: Heterogenous federated learning via model distillation. *arXiv preprint arXiv:1910.03581*.

[Li et al., 2020] Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. (2020). Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450.

[Li et al., 2019] Li, X., Huang, K., Yang, W., Wang, S., and Zhang, Z. (2019). On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*.

[Lin et al., 2020] Lin, T., Kong, L., Stich, S. U., and Jaggi, M. (2020). Ensemble distillation for robust model fusion in federated learning. *Advances in Neural Information Processing Systems*, 33:2351–2363.

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

[McMahan et al., 2017] McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR.

[Mnih et al., 2013] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

[Paszke et al., 2017] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch. In *NIPS 2017 Workshop on Autodiff*.

[Paszke et al., 2019] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

[Sandler et al., 2018] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520.

[Sattler et al., 2020a] Sattler, F., Müller, K.-R., and Samek, W. (2020a). Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE transactions on neural networks and learning systems*, 32(8):3710–3722.

[Sattler et al., 2020b] Sattler, F., Müller, K.-R., Wiegand, T., and Samek, W. (2020b). On the byzantine robustness of clustered federated learning. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8861–8865. IEEE.

[Sattler et al., 2019] Sattler, F., Wiedemann, S., Müller, K.-R., and Samek, W. (2019). Robust and communication-efficient federated learning from non-iid data. *IEEE transactions on neural networks and learning systems*, 31(9):3400–3413.

[Smith et al., 2017] Smith, V., Chiang, C.-K., Sanjabi, M., and Talwalkar, A. S. (2017). Federated multi-task learning. *Advances in neural information processing systems*, 30.

[Tan et al., 2022] Tan, A. Z., Yu, H., Cui, L., and Yang, Q. (2022). Towards personalized federated learning. *IEEE Transactions on Neural Networks and Learning Systems*.

[Wang et al., 2020] Wang, H., Kaplan, Z., Niu, D., and Li, B. (2020). Optimizing federated learning on non-iid data with reinforcement learning. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 1698–1707. IEEE.

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

[Yang et al., 2021] Yang, M., Wang, X., Zhu, H., Wang, H., and Qian, H. (2021). Federated learning with class imbalance reduction. In *2021 29th European Signal Processing Conference (EUSIPCO)*, pages 2174–2178. IEEE.

[Yurochkin et al., 2019] Yurochkin, M., Agarwal, M., Ghosh, S., Greenewald, K., Hoang, N., and Khazaeni, Y. (2019). Bayesian nonparametric federated learning of neural networks. In *International Conference on Machine Learning*, pages 7252–7261. PMLR.

[Zhang et al., 2020] Zhang, M., Sapra, K., Fidler, S., Yeung, S., and Alvarez, J. M. (2020). Personalized federated learning with first order model optimization. *arXiv preprint arXiv:2012.08565*.

[Zhang et al., 2021] Zhang, W., Wang, X., Zhou, P., Wu, W., and Zhang, X. (2021). Client selection for federated learning with non-iid data in mobile edge computing. *IEEE Access*, 9:24462–24474.

[Zhang and Yang, 2018] Zhang, Y. and Yang, Q. (2018). An overview of multi-task learning. *National Science Review*, 5(1):30–43.

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

[Zhu et al., 2021] Zhu, H., Xu, J., Liu, S., and Jin, Y. (2021). Federated learning on non-iid data: A survey. *Neurocomputing*, 465:371–390.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Not Applicable]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]