FedProto: Federated Prototype Learning across Heterogeneous Clients

Yue Tan[1], Guodong Long[1], Lu Liu[1], Tianyi Zhou[2,3], Qinghua Lu[4], Jing Jiang[1], Chengqi Zhang [1]

[1] Australian Artificial Intelligence Institute, FEIT, University of Technology Sydney
[2] University of Washington
[3] University of Maryland, College Park
[4] Data61, CSIRO
yue.tan@student.uts.edu.au, {guodong.long, jing.jiang, chengqi.zhang}@uts.edu.au,
lu.liu.cs@icloud.com, tianyizh@uw.edu, qinghua.lu@data61.csiro.au

Abstract

Heterogeneity across clients in federated learning (FL) usually hinders the optimization convergence and generalization performance when the aggregation of clients' knowledge occurs in the gradient space. For example, clients may differ in terms of data distribution, network latency, input/output space, and/or model architecture, which can easily lead to the misalignment of their local gradients. To improve the tolerance to heterogeneity, we propose a novel federated prototype learning (`FedProto`) framework in which the clients and server communicate the abstract class prototypes instead of the gradients. `FedProto` aggregates the local prototypes collected from different clients, and then sends the global prototypes back to all clients to regularize the training of local models. The training on each client aims to minimize the classification error on the local data while keeping the resulting local prototypes sufficiently close to the corresponding global ones. Moreover, we provide a theoretical analysis to the convergence rate of `FedProto` under non-convex objectives. In experiments, we propose a benchmark setting tailored for heterogeneous FL, with `FedProto` outperforming several recent FL approaches on multiple datasets.

Introduction

Federated learning (FL) is widely used in multiple applications to enable collaborative learning across a variety of clients without sharing private data. It aims at training a global model on a centralized server while all data are distributed over many local clients and cannot be freely transmitted for privacy or communication concerns (McMahan et al. 2017). The iterative process of FL has two steps: (1) each local client is synchronized by the global model and then trained using its local data; and (2) the server updates the global model by aggregating all the local models. Considering that the model aggregation occurs in the gradient space, traditional FL still has some practical challenges caused by the heterogeneity of *data* and *model* (Kairouz et al. 2019). Efficient algorithms suitable to overcome both these two challenges have not yet been fully developed or systematically examined.

To tackle the *statistical heterogeneity* of data distributions, one straightforward solution is to maintain multiple global models for different local distributions, e.g., the works for clustered FL (Sattler, Müller, and Samek 2020). Another widely studied strategy is personalized FL (Tan et al. 2021) where a personalized model is generated for each client by leveraging both global and local information. Nevertheless, most of these methods depend on gradient-based aggregation, resulting in high communication costs and heavy reliance on homogeneous local models.

However, in real-world applications, *model heterogeneity* is common because of varying hardware and computation capabilities across clients (Long et al. 2020). Knowledge Distillation (KD)-based FL (Lin et al. 2020) addresses this challenge by transferring the teacher model's knowledge to student models with different model architectures. However, these methods require an extra public dataset to align the student and teacher models' outputs, increasing the computation costs. Moreover, the performance of KD-based FL can significantly degrade with the increase in the distribution divergence between the public dataset and on-client datasets that are usually non-IID.

Inspired by prototype learning, merging the prototypes over heterogeneous datasets can effectively integrate the feature representations from diverse data distributions (Snell, Swersky, and Zemel 2017; Liu et al. 2020; Dvornik, Schmid, and Mairal 2020). On-client intelligent agents in the FL system can share knowledge by exchanging information in terms of representations, despite statistical and model heterogeneity. For example, when we talk about "dog", different people will have a unique "imagined picture" or "prototype" to represent the concept "dog". Their prototypes may be slightly diverse due to different life experience and visual memory. Exchanging these concept-specific prototypes across people enables them to acquire more knowledge about the concept "dog". Treating each FL client as a human-like intelligent agent, the core idea of our method is to exchange prototypes rather than share model parameters or raw data, which can naturally match the knowledge acquisition behavior of humans.

In this paper, we propose a novel prototype aggregation-based FL framework where only prototypes are transmitted between the server and clients. The proposed solution does not require model parameters or gradients to be aggregated, so it has a huge potential to be a robust framework for various heterogeneous FL scenarios. Concretely, each client can have different model architectures and input/output spaces,

but they can still exchange information by sharing prototypes. Each abstract prototype represents a class by the mean representations transformed from the observed samples belonging to the same class. Aggregating the prototypes allows for efficient communication across heterogeneous clients.

Our main contributions can be summarized as follows:

- We propose a benchmark setting tailored for heterogeneous FL that considers a more general heterogeneous scenario across local clients.
- We present a novel FL method that significantly improves the communication efficiency in the heterogeneous setting. To the best of our knowledge, we are the first to propose prototype aggregation-based FL.
- We theoretically provide a convergence guarantee for our method and carefully derive the convergence rate under non-convex conditions.
- Extensive experiments show the superiority of our proposed method in terms of communication efficiency and test performance in several benchmark datasets.

Related Work

Heterogeneous Federated Learning

Statistical heterogeneity across clients (also known as the non-IID problem) is the most important challenge of FL. `FedProx` (Li et al. 2020) proposed a local regularization term to optimize each client's local model. Some recent studies (Arivazhagan et al. 2019; Liang et al. 2020; Deng, Kamani, and Mahdavi 2020) train personalized models to leverage both globally shared information and the personalized part (Tan et al. 2021). The third solution is to provide multiple global models by clustering the local models (Mansour et al. 2020; Ghosh et al. 2020; Sattler, Müller, and Samek 2020) into multiple groups or clusters. Recently, self-supervised learning strategies are incorporated into the local training phase to handle the heterogeneity challenges (Li, He, and Song 2021; Liu et al. 2021a). (Fallah, Mokhtari, and Ozdaglar 2020) applies meta-training strategy for personalized FL.

Heterogeneous model architecture is another major challenging scenario of FL. The recently proposed KD-based FL (Lin et al. 2020; Jeong et al. 2018; Li and Wang 2020; Long et al. 2021) can serve as an alternative solution to address this challenge. In particular, with the assumption of adding a shared toy dataset in the federated setting, these KD-based FL methods can distill knowledge from a teacher model to student models with different model architectures. Some recent studies have also attempted to combine the neural architecture search with federated learning (Zhu, Zhang, and Jin 2020; He, Annavaram, and Avestimehr 2020; Singh et al. 2020), which can be applied to discover a customized model architecture for each group of clients with different hardware capabilities and configurations. A collective learning platform is proposed to handle heterogeneous architectures without access to the local training data and architectures in (Hoang et al. 2019). Moreover, functionality-based neural matching across local models (Wang et al. 2020a) can aggregate neurons with similar functionality regardless of the variance of the model architectures.

However, most of these mentioned FL methods focus on only one heterogeneous challenging scenario. All of them use gradient-based aggregation methods which will raise concerns about communication efficiency and gradient-based attacks (Zhu, Liu, and Han 2019; Chen et al. 2020; Liu et al. 2021b).

Prototype Learning

The concept of prototypes (the mean of multiple features) has been explored in a variety of tasks. In image classification, a prototype can be a proxy of a class and is calculated as the mean of the feature vectors within every class (Snell, Swersky, and Zemel 2017). In action recognition, the features of a video in different timestamps can be averaged to serve as the representation of the video (Simonyan and Zisserman 2014). Aggregated local features can serve as descriptors for image retrieval (Babenko and Lempitsky 2015). Averaging word embeddings as the representation of a sentence can achieve competitive performance on multiple NLP benchmarks (Wieting et al. 2015). The authors in (Hoang et al. 2020) use prototypes to represent task-agnostic information in distributed machine learning and propose a new fusion paradigm to integrate those prototypes to generate a new model for a new task. In (Michieli and Ozay 2021), prototype margins are used to optimize visual feature representations for FL. In our paper, we borrow the concept of prototypes to represent one class and apply prototype aggregation in the setting of heterogeneous FL.

In general, prototypes are widely used in learning scenarios with a limited number of training samples (Snell, Swersky, and Zemel 2017). This learning scenario is consistent with the latent assumption of cross-client FL: that each client has a limited number of instances to independently train a model with the desired performance. The assumption has been widely supported by the FL-based benchmark datasets (Caldas et al. 2018; He et al. 2020) and in related applications, such as healthcare (Rieke et al. 2020; Xu et al. 2020) and street image object detection (Luo et al. 2019).

Problem Setting

Heterogeneous Federated Learning Setting

In federated learning, each client owns a local private dataset D_i drawn from distribution $\mathbb{P}_i(x, y)$, where x and y denote the input features and corresponding class labels, respectively. Usually, clients share a model $\mathcal{F}(\omega; x)$ with the same architecture and hyperparameters. This model is parameterized by learnable weights ω and input features x. The objective function of `FedAvg` (McMahan et al. 2017) is:

$$\arg\min_{\omega} \sum_{i=1}^{m} \frac{|D_i|}{N} \mathcal{L}_S(\mathcal{F}(\omega; x), y), \tag{1}$$

where ω is the global model's parameters, m denotes the number of clients, N is the total number of instances over all clients, $\mathcal{F}$ is the shared model, and $\mathcal{L}_S$ is a general definition of any supervised learning task (e.g., a cross-entropy loss).

However, in a real-world FL environment, each client may represent a mobile phone with a specific user behavior pattern or a sensor deployed in a particular location, leading to

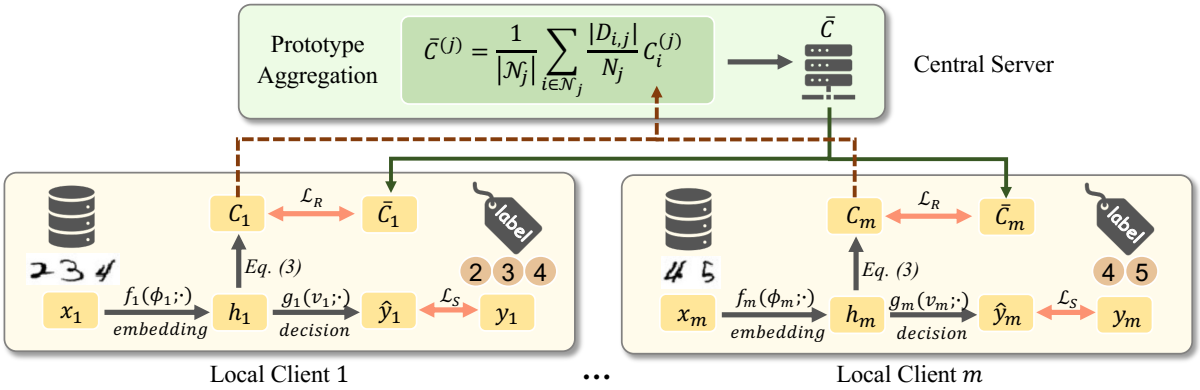


Figure 1: An overview of FedProto in the heterogeneous setting. For example, the first client is to recognize the digits $2, 3, 4$, while the m-th client is to recognize the digits $4, 5$. First, the clients update their local prototype sets by minimizing the loss of classification error $\mathcal{L}_S$ and the distance between global prototypes and local prototypes $\mathcal{L}_R$. Then, the clients send their prototypes to the central server. The central server generates global prototypes and returns them to all clients to regularize the training of local models.

statistical and/or model heterogeneous environment. In the statistical heterogeneity setting, $\mathbb{P}_i$ varies across clients, indicating heterogeneous input/output space for x and y. For example, $\mathbb{P}_i$ on different clients can be the data distributions over different subsets of classes. In the model heterogeneity setting, $\mathcal{F}_i$ varies across clients, indicating different model architectures and hyperparameters. For the i-th client, the training procedure is to minimize the loss as defined below:

$$\arg\min_{\omega_1,\omega_2,\ldots,\omega_m} \sum_{i=1}^{m} \frac{|D_i|}{N} \mathcal{L}_S(\mathcal{F}_i(\omega_i; x), y). \tag{2}$$

Most existing methods cannot well handle the heterogeneous settings above. In particular, the fact that $\mathcal{F}_i$ has a different model architecture would cause ω_i to have a different format and size. Thus, the global model's parameter ω cannot be optimized by averaging ω_i. To tackle this challenge, we propose to communicate and aggregate prototypes in FL.

Prototype-Based Aggregation Setting

Heterogeneous FL focuses on the robustness to tackle heterogeneous input/output spaces, distributions and model architectures. For example, the datasets D_i and D_k on two clients i and k may take different statistical distributions of labels. This is common for a photo classification APP installed on mobile clients, where the server needs to recognize many classes $\mathbb{C} = \{C^{(1)}, C^{(2)}, \ldots\}$, while each client only needs to recognize a few classes that constitute a subset of $\mathbb{C}$. The class set can vary across clients, though there are overlaps.

In general, the deep learning-based models comprise two parts: (1) representation layers (a.k.a. embedding functions) to transform the input from the original feature space to the embedding space; and (2) decision layers to make a classification decision for a given learning task.

Representation layers The embedding function of the i-th client is $f_i(\phi_i)$ parameterized by ϕ_i. We denote $h_i = f_i(\phi_i; x)$ as the embeddings of x.

Decision layers Given a supervised learning task, a prediction for x can be generated by the function $g_i(\nu_i)$ parameterized by ν_i. So, the labelling function can be written as $\mathcal{F}_i(\phi_i, \nu_i) = g_i(\nu_i) \circ f_i(\phi_i)$, and we use ω_i to represent (ϕ_i, ν_i) for short.

Prototype We define a prototype $C^{(j)}$ to represent the j-th class in $\mathbb{C}$. For the i-th client, the prototype is the mean value of the embedding vectors of instances in class j,

$$C_i^{(j)} = \frac{1}{|D_{i,j}|} \sum_{(x,y) \in D_{i,j}} f_i(\phi_i; x), \tag{3}$$

where $D_{i,j}$, a subset of the local dataset D_i, is comprised of training instances belonging to the j-th class.

Prototype-based model inference In the inference stage of the learning task, we can simply predict the label $\hat{y}$ to an instance x by measuring the L2 distance between the instance's representational vector $f(\phi; x)$ and the prototype $C^{(j)}$ as follows:

$$\hat{y} = \arg\min_j ||f(\phi; x) - C^{(j)}||_2. \tag{4}$$

Methodology

We propose a solution for heterogeneous FL that uses prototypes as the key component for exchanging information across the server and the clients.

An overview of the proposed framework is shown in Figure 1. The central server receives local prototype sets $C_1, C_2, \ldots, C_m$ from m local clients, and then aggregates the prototypes by averaging them. In the heterogeneous FL setting, these prototype sets overlap but are not the same. Taking the MNIST dataset as an example, the first client is to recognize the digits $2, 3, 4$, while another client is to recognize the digits $4, 5$. These are two different handwritten digits set; nonetheless, there is an overlap. The server automatically aggregates prototypes from the overlapping class space across the clients.

Using prototypes in FL, we do not need to exchange gradients or model parameters, which means that the proposed solution can tackle heterogeneous model architectures. Moreover, the prototype-based FL does not require each client to provide the same classes, meaning the heterogeneous class spaces are well supported. Thus, heterogeneity challenges in FL can be addressed.

Optimization Objective

The objective of `FedProto` is to solve a joint optimization problem on a distributed network. `FedProto` applies prototype-based communication, which allows a local model to align its prototypes with other local models while minimizing the sum of loss for all clients' local learning tasks. The objective of federated prototype learning across heterogeneous clients can be formulated as

$$\begin{aligned}\arg\min_{\{\bar{C}^{(j)}\}_{j=1}^{|\mathbb{C}|}} \sum_{i=1}^{m} \frac{|D_i|}{N} \mathcal{L}_S(\mathcal{F}_i(\omega_i; x), y) + \\ \lambda \cdot \sum_{j=1}^{|\mathbb{C}|} \sum_{i=1}^{m} \frac{|D_{i,j}|}{N_j} \mathcal{L}_R(\bar{C}_i^{(j)}, C_i^{(j)}),\end{aligned} \tag{5}$$

where $\mathcal{L}_S$ is the loss of supervised learning (as defined in Eq. (2)) and $\mathcal{L}_R$ is a regularization term that measures the distance (we use L2 distance) between a local prototype $C^{(j)}$ and the corresponding global prototypes $\bar{C}^{(j)}$.

N is the total number of instances over all clients, and N_j is the number of instances belonging to class j over all clients.

The optimization problem can be addressed by alternate minimization that iterates the following two steps: (1) minimization w.r.t. each ω_i with $\bar{C}_i^{(j)}$ fixed; and (2) minimization w.r.t. $\bar{C}_i^{(j)}$ with all ω_i fixed. In a distributed setting, step (1) reduces to conventional supervised learning on each client using its local data, while step (2) aggregates local prototypes from local clients on the server end. Further details concerning these two steps can be seen in Algorithm 1.

Global Prototype Aggregation

Given the data and model heterogeneity in the participating clients, the optimal model parameters for each client are not the same. This means that gradient-based communication cannot sufficiently provide useful information to each client. However, the same label space allows the participating clients to share the same embedding space and information can be efficiently exchanged across heterogeneous clients by aggregating prototypes according to the classes they belong to.

Given a class j, the server receives prototypes from a set of clients that have class j. A global prototype $\bar{C}^{(j)}$ for class j is generated after the prototype aggregating operation,

$$\bar{C}^{(j)} = \frac{1}{|\mathcal{N}_j|} \sum_{i \in \mathcal{N}_j} \frac{|D_{i,j}|}{N_j} C_i^{(j)}, \tag{6}$$

where $C_i^{(j)}$ denotes the prototype of class j from client i, and $\mathcal{N}_j$ denotes the set of clients that have class j.

Algorithm 1: `FedProto`

Input: $D_i, \omega_i, i = 1, \cdots, m$
Server executes:
1: Initialize global prototype set $\{\bar{C}^{(j)}\}$ for all classes.
2: **for** each round $T = 1, 2, \ldots$ **do**
3: **for** each client i **in parallel do**
4: $C_i \leftarrow$ LocalUpdate $(i, \bar{C}_i)$
5: **end for**
6: Update global prototype by Eq. 6.
7: Update local prototype set C_i with prototypes in $\{\bar{C}^{(j)}\}$
8: **end for**

LocalUpdate$(i, \bar{C}_i)$:
1: **for** each local epoch **do**
2: **for** batch $(x_i, y_i) \in D_i$ **do**
3: Compute local prototype by Eq. 3.
4: Compute loss by Eq. 7 using local prototypes.
5: Update local model according to the loss.
6: **end for**
7: **end for**
8: **return** $C^{(i)}$

Local Model Update

The client needs to update the local model to generate a consistent prototype across the clients. To this end, a regularization term is added to the local loss function, enabling the local prototypes $C_i^{(j)}$ to approach global prototypes $\bar{C}_i^{(j)}$ while minimizing the loss of the classification error. In particular, the loss function is defined as follows:

$$\mathcal{L}(D_i, \omega_i) = \mathcal{L}_S(\mathcal{F}_i(\omega_i; x), y) + \lambda \cdot \mathcal{L}_R\left(\bar{C}_i, C_i\right), \tag{7}$$

where λ is an importance weight, and $\mathcal{L}_R$ is the regularization term that can be defined as:

$$\mathcal{L}_R = \sum_j d(C_i^{(j)}, \bar{C}_i^{(j)}), \tag{8}$$

where d is a distance metric of local generated prototypes $C^{(j)}$ and global aggregated prototypes $\bar{C}^{(j)}$. The distance measurement can take a variety of forms, such as L1 distance, L2 distance, and earth mover's distance.

Convergence Analysis

We provide insights into the convergence analysis for `FedProto`. We denote the local objective function defined in Eq. 7 as $\mathcal{L}$ with a subscript indicating the number of iterations and make the following assumptions similar to existing general frameworks (Wang et al. 2020b; Li et al. 2020).

Assumption 1. (Lipschitz Smooth). *Each local objective function is L_1-Lipschitz smooth, which means that the gradient of local objective function is L_1-Lipschitz continuous,*

$$\begin{aligned}\|\nabla \mathcal{L}_{t_1} - \nabla \mathcal{L}_{t_2}\|_2 \leq L_1 \|\omega_{i,t_1} - \omega_{i,t_2}\|_2, \\ \forall t_1, t_2 > 0, i \in \{1, 2, \ldots, m\}.\end{aligned} \tag{9}$$

This also implies the following quadratic bound,

$$\begin{aligned}\mathcal{L}_{t_1} - \mathcal{L}_{t_2} \leq \langle \nabla \mathcal{L}_{t_2}, (\omega_{i,t_1} - \omega_{i,t_2}) \rangle + \frac{L_1}{2} \|\omega_{i,t_1} - \omega_{i,t_2}\|_2^2, \\ \forall t_1, t_2 > 0, \quad i \in \{1, 2, \ldots, m\}.\end{aligned} \tag{10}$$

Assumption 2. (Unbiased Gradient and Bounded Variance). *The stochastic gradient $g_{i,t} = \nabla\mathcal{L}(\omega_t, \xi_t)$ is an unbiased estimator of the local gradient for each client. Suppose its expectation*

$$\mathbb{E}_{\xi_i \sim D_i}[g_{i,t}] = \nabla\mathcal{L}(\omega_{i,t}) = \nabla\mathcal{L}_t, \forall i \in \{1, 2, \ldots, m\}, \quad (11)$$

and its variance is bounded by σ^2:

$$\mathbb{E}[\|g_{i,t} - \nabla\mathcal{L}(\omega_{i,t})\|_2^2] \leq \sigma^2, \forall i \in \{1, 2, \ldots, m\}, \sigma^2 \geq 0. \quad (12)$$

Assumption 3. (Bounded Expectation of Euclidean norm of Stochastic Gradients).*The expectation of the stochastic gradient is bounded by G:*

$$\mathbb{E}[\|g_{i,t}\|_2] \leq G, \forall i \in \{1, 2, \ldots, m\}. \quad (13)$$

Assumption 4. (Lipschitz Continuity). *Each local embedding function is L_2-Lipschitz continuous, that is,*

$$\begin{aligned}\|f_i(\phi_{i,t_1}) - f_i(\phi_{i,t_2})\| \leq L_2\|\phi_{i,t_1} - \phi_{i,t_2}\|_2,\\ \forall t_1, t_2 > 0, i \in \{1, 2, \ldots, m\}.\end{aligned} \quad (14)$$

Based on the above assumptions, we present the theoretical results for the non-convex problem. The expected decrease per round is given in Theorem 1. We denote $e \in \{1/2, 1, 2, \ldots, E\}$ as the local iteration, and t as the global communication round. Moreover, tE represents the time step before prototype aggregation, and $tE + 1/2$ represents the time step between prototype aggregation and the first iteration of the current round.

Theorem 1. (One-round deviation). *Let Assumption 1 to 4 hold. For an arbitrary client, after every communication round, we have,*

$$\begin{aligned}\mathbb{E}[\mathcal{L}_{(t+1)E+1/2}] \leq &\mathcal{L}_{tE+1/2} - \left(\eta - \frac{L_1\eta^2}{2}\right)\sum_{e=1/2}^{E-1}\|\nabla\mathcal{L}_{tE+e}\|_2^2\\ &+ \frac{L_1 E\eta^2}{2}\sigma^2 + \lambda L_2\eta EG.\end{aligned} \quad (15)$$

Theorem 1 indicates the deviation bound of the local objective function for an arbitrary client after each communication round. Convergence can be guaranteed when there is a certain expected one-round decrease, which can be achieved by choosing appropriate η and λ.

Corollary 1. (Non-convex `FedProto` convergence). *The loss function $\mathcal{L}$ of an arbitrary client monotonously decreases in every communication round when*

$$\eta_{e'} < \frac{2(\sum_{e=1/2}^{e'}\|\nabla\mathcal{L}_{tE+e}\|_2^2 - \lambda L_2 EG)}{L_1(\sum_{e=1/2}^{e'}\|\nabla\mathcal{L}_{tE+e}\|_2^2 + E\sigma^2)}, \quad (16)$$

where $e' = 1/2, 1, \ldots, E-1$, and

$$\lambda_t < \frac{\|\nabla\mathcal{L}_{tE+1/2}\|_2^2}{L_2 EG}. \quad (17)$$

Thus, the loss function converges.

Corollary 1 is to ensure the expected deviation of $\mathcal{L}$ to be negative, so the loss function converges. It can guide the choice of appropriate values for the learning rate η and the importance weight λ to guarantee the convergence.

Theorem 2. (Non-convex convergence rate of `FedProto`). *Let Assumption 1 to 4 hold and $\Delta = \mathcal{L}_0 - \mathcal{L}^*$ where $\mathcal{L}^*$ refers to the local optimum. For an arbitrary client, given any $\epsilon > 0$, after*

$$T = \frac{2\Delta}{E\epsilon(2\eta - L_1\eta^2) - E\eta(L_1\eta\sigma^2 + 2\lambda L_2 G)} \quad (18)$$

communication rounds of `FedProto`, we have

$$\frac{1}{TE}\sum_{t=0}^{T-1}\sum_{e=1/2}^{E-1}\mathbb{E}[\|\nabla\mathcal{L}_{tE+e}\|_2^2] < \epsilon, \quad (19)$$

if

$$\eta < \frac{2(\epsilon - \lambda L_2 G)}{L_1(\epsilon + \sigma^2)} \text{ and } \lambda < \frac{\epsilon}{L_2 G}.$$

Theorem 2 provides the convergence rate, which can confine the expected L2-norm of gradients to any bound, denoted as ϵ, after carefully selecting the number of communication rounds T and hyperparameters including η and λ. The smaller ϵ is, the larger T is, which means that the tighter the bound is, more communication rounds is required. A detailed proof and analysis are given in Appendix B.

Discussion

In this section, we discuss the superiority of `FedProto` from three perspectives: model inference, communication efficiency, and privacy preserving.

Model Inference

Unlike many FL methods, the global model in FedProto is not a classifier but a set of class prototypes. When a new client is added to the network, one can initialize its local model with the representation layers of a pre-trained model, e.g. a ResNet18 on ImageNet, and random decision layers. Then, the local client will download the global prototypes of the classes covered in its local dataset and fine-tune the local model by minimizing the local objective. This can support new clients with novel model architectures and spend less time fine-tuning the model on heterogeneous datasets.

Communication Efficiency

Our proposed method only transmits prototypes between the server and clients. In general, the size of the prototypes is usually much smaller than the size of the model parameters. Taking MNIST as an example, the prototype size is 50 for each class, while the number of model parameters is 21,500. More details can be found in the experimental section.

Privacy Preserving

The proposed `FedProto` requires the exchange of prototypes rather than model parameters between the server and the clients. This property brings benefits to FL in terms of privacy preserving. First, prototypes naturally protect the data privacy, because they are 1D-vectors generated by averaging the low-dimension representations of samples from the same class, which is an irreversible process. Second, attackers cannot reconstruct raw data from prototypes without the access to local models. Moreover, `FedProto` can be integrated with various privacy-preserving techniques to further enhance the reliability of the system.

Dataset	Method	Stdev	Test Average Acc			# of Comm Rounds	# of Comm Params ($\times 10^3$)
			$n=3$	$n=4$	$n=5$		
MNIST	`Local`	2	94.05±2.93	93.35±3.26	92.92±3.17	0	0
	`FeSEM` (Xie et al. 2020)	2	95.26±3.48	**97.06±2.72**	96.31±2.41	150	430
	`FedProx` (Li et al. 2020)	2	96.26±2.89	96.40±3.33	95.65±3.38	110	430
	`FedPer` (Arivazhagan et al. 2019)	2	95.57±2.96	96.44±2.62	95.55±3.13	100	106
	`FedAvg` (McMahan et al. 2017)	2	95.04±6.48	94.32±4.89	93.22±4.39	150	430
	`FedRep` (Collins et al. 2021)	2	94.96±2.78	95.18±3.80	94.94±2.81	100	110
	`FedProto`	2	**97.13±0.30**	96.80±0.41	**96.70±0.29**	100	**4**
	`FedProto-mh`	2	97.07±0.50	96.65±0.31	96.22±0.36	100	**4**
FEMNIST	`Local`	1	92.50±10.42	91.16±5.64	87.91±8.44	0	0
	`FeSEM` (Xie et al. 2020)	1	93.39±6.75	91.06±6.43	89.61±7.89	200	16,000
	`FedProx` (Li et al. 2020)	1	94.53±5.33	90.71±6.24	91.33±7.32	300	16,000
	`FedPer` (Arivazhagan et al. 2019)	1	93.47±5.44	90.22±7.63	87.73±9.64	250	102
	`FedAvg` (McMahan et al. 2017)	1	94.50±5.29	91.39±5.23	90.95±7.22	300	16,000
	`FedRep` (Collins et al. 2021)	1	93.36±5.34	91.41±5.89	89.98±6.88	200	102
	`FedProto`	1	96.82±1.75	**94.93±1.61**	93.67±2.23	120	**4**
	`FedProto-mh`	1	**97.10±1.63**	94.83±1.60	**93.76±2.30**	120	**4**
CIFAR10	`Local`	1	79.72±9.45	67.62±7.15	58.64±6.57	0	0
	`FeSEM` (Xie et al. 2020)	1	80.19±3.31	76.40±3.23	74.17±3.51	120	235,000
	`FedProx` (Li et al. 2020)	1	83.25±2.44	79.20±1.31	76.19±2.23	150	235,000
	`FedPer` (Arivazhagan et al. 2019)	1	84.38±4.58	78.73±4.59	76.21±4.27	130	225,000
	`FedAvg` (McMahan et al. 2017)	1	81.72±2.77	76.77±2.37	75.74±2.61	150	235,000
	`FedRep` (Collins et al. 2021)	1	81.44±10.48	76.93±7.46	73.36±7.04	110	225,000
	`FedProto`	1	**84.49±1.97**	79.12±2.03	**77.08±1.98**	110	**41**
	`FedProto-mh`	1	83.63±1.60	**79.49±1.78**	76.94±1.33	110	**41**

Table 1: Comparison of FL methods on three benchmark datasets with non-IID split over clients. The best results are in bold. It appears that `FedProto`, compared to baselines, achieves higher accuracy while using much fewer communicated parameters.

Experiments

Training Setups

Datasets and local models We implement the typical federated setting where each client owns its local data and transmits/receives information to/from the central server. We use three popular benchmark datasets: MNIST (LeCun 1998), FEMNIST (Caldas et al. 2018) and CIFAR10 (Krizhevsky 2012). We consider a multi-layer CNN which consists of 2 convolutional layers then 2 fully connected layers for both MNIST and FEMNIST, and ResNet18 (He et al. 2016) for CIFAR10.

Local tasks Each client learns a supervised learning task. In particular, to illustrate the local task, we borrow the concept of n-way k-shot from few-shot learning where n controls the number of classes and k controls the number of training instances per class. To mimic the heterogeneous scenario, we randomly change the value of n and k in different clients. We define an average value for n and k, and then add a random noise to each user's n as well as k. The purpose of the variance of n is to control the heterogeneity of the class space, while the purpose of the variance of k is to control the imbalance in data size.

Baselines of FL We study the performance of `FedProto` under both the statistical and model heterogeneous settings (`FedProto-mh`) and make comparisons with baselines, including `Local` where an individual model is trained for each client without any communication with others, `FedAvg` (McMahan et al. 2017), `FedProx` (Li et al. 2020), `FeSEM` (Xie et al. 2020), `FedPer` (Arivazhagan et al. 2019), and `FedRep` (Collins et al. 2021).

Implementation Details We implement `FedProto` and the baseline methods in PyTorch. We use 20 clients for all datasets and all clients are sampled in each communication round. The average size of each class in each client is set to be 100. For MNIST and FEMNIST dataset, our initial set of hyperparameters was taken directly from the default set of hyperparamters in (McMahan et al. 2017). For CIFAR10, ResNet18 pre-trained on ImageNet (Krizhevsky, Sutskever, and Hinton 2017) is used as the initial model. The initial average test accuracy of the pre-trained network on CIFAR10 is 27.55%. A detailed setup including the choice of hyperparameters is given in Appendix A.

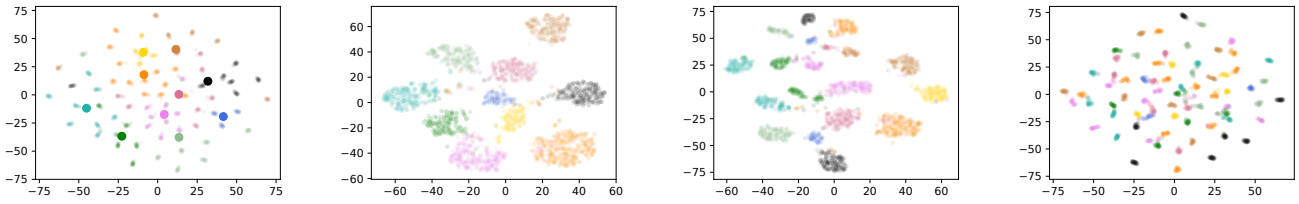
Performance in Non-IID Federated Setting

We compare `FedProto` with other baseline methods that are either classical FL methods or FL methods with an emphasis on statistical heterogeneity. All methods are adapted to fit this heterogeneous setting.

Statistical heterogeneity simulations In our setting, we assume that all clients perform learning tasks with heterogeneous statistical distributions. In order to simulate different levels of heterogeneity, we fix the standard deviation to be 1 or 2, aiming to create heterogeneity in both class spaces and data sizes, which is common in real-world scenarios.

Model heterogeneity simulations For the model heterogeneous setting, we consider minor differences in model architectures across clients. In MNIST and FEMNIST, the number of output channels in the convolutional layers is set to either 18, 20 or 22, while in CIFAR10, the stride of convolutional layers is set differently across different clients. This kind of model heterogeneity brings about challenges for model parameter averaging because the parameters in different clients are not always the same size.

The average test accuracy over all clients is reported in



(a) FedProto (b) FedAvg (c) FeSEM (d) FedPer

Figure 2: t-SNE visualization of the samples and/or prototypes produced by FedProto and other FL methods. We consider 20 clients for MNIST. The average number of classes per client is $n = 3$. (a) FedProto: Samples within the same class have multiple centers with each center representing local prototype of a client. The global prototype is the central point of the class samples. (b) FedAvg: Samples within the same class cluster in the same area. (c) FeSEM: Samples within the same class gather in several clusters according to the algorithm parameters. (d) FedPer: Each cluster indicates one class in a specific client.

Table 1. It can be seen that FedProto achieves the highest accuracy and the least variance in most cases, ensuring uniformity among heterogeneous clients.

Communication efficiency Communication costs have always been posed as a challenge in FL, considering several limitations in existing communication channels. Therefore, we also report the number of communication rounds required for convergence and the number of parameters communicated per round in Table 1. It can be seen that the number of parameters communicated per round in FedProto is much lower than in the case of FedAvg. Furthermore, FedProto requires the fewest communication rounds for the local optimization. This suggests that when the heterogeneity level is high across the clients, sharing more parameters does not always lead to better results. It is more important to identify which part to share in order to benefit the current system to a great extent. More performance results are shown in Appendix A.

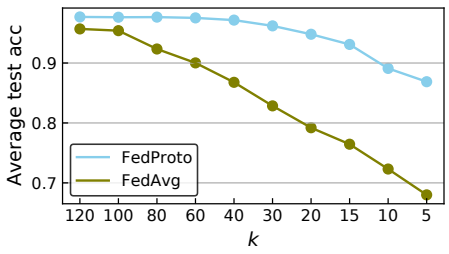


Figure 3: Average test accuracy of FedProto and FedAvg on MNIST with varying number of samples in each class.

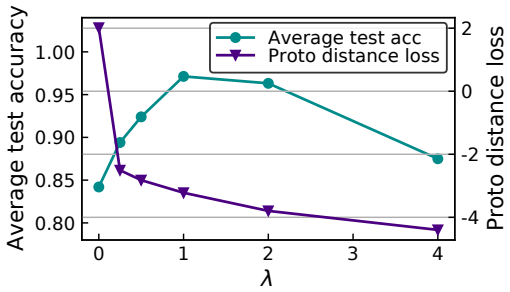


Figure 4: Average test accuracy on FEMNIST under varying importance weight λ.

Visualization of prototypes achieved by FedProto We visualize the samples in MNIST test set by t-SNE (Van der Maaten and Hinton 2008). In Figure 2(a), small points in different colors represent samples in different classes, with large points representing corresponding global prototypes. In Figure 2(b), 2(c) and 2(d), the points in different colors refer to the representations of samples belonging to different classes. Better generalization means that there are more samples within the same class cluster in the same area, which can be achieved in a centralized setting, while better personalization means that it is easier to determine to which client the samples belong. It can be seen that samples within the same class but from various clients are close but separable in FedProto. This indicates that FedProto is more successful in achieving the balance between generalization and personalization, while other methods lacks either the generalization or the personalization ability.

Scalability of FedProto on varying number of samples Figure 3 shows that FedProto can scale to scenarios with fewer samples available on clients. The test accuracy consistently decreases when there are fewer samples for training, but FedProto drops more slowly than FedAvg as a result of its adaptability and scalability on various data sizes.

FedProto under varying λ Figure 4 shows the varying performance under different values of λ in Eq. (5). We tried a set of values selected from $[0, 4]$ and reported the average test accuracy and proto distance loss with n=3, k=100 in FEMNIST dataset. The best value of λ is 1 in this scenario. As λ increases, the proto distance loss (regularization term) decreases, while the average test accuracy experiences a sharp rise from λ=0 to λ=1 before a drop in the number of 6%, demonstrating the efficacy of prototype aggregation.

Conclusion

In this paper, we propose a novel prototype aggregation-based FL method to tackle challenging FL scenarios with heterogeneous input/output spaces, data distributions, and model architectures. The proposed method collaboratively trains intelligent models by exchanging prototypes rather than gradients, which offers new insights for designing prototype-based FL. The effectiveness of the proposed method has been comprehensively analyzed from both theoretical and experimental perspectives.

References

Arivazhagan, M. G.; Aggarwal, V.; Singh, A. K.; and Choudhary, S. 2019. Federated learning with personalization layers. *arXiv preprint arXiv:1912.00818*.

Babenko, A.; and Lempitsky, V. 2015. Aggregating local deep features for image retrieval. In *Proceedings of the IEEE international conference on computer vision*, 1269–1277.

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: 1812.01097*.

Chen, C.; Zhang, J.; Tung, A. K.; Kankanhalli, M.; and Chen, G. 2020. Robust federated recommendation system. *arXiv preprint arXiv:2006.08259*.

Collins, L.; Hassani, H.; Mokhtari, A.; and Shakkottai, S. 2021. Exploiting Shared Representations for Personalized Federated Learning. *International Conference on Machine Learning*.

Deng, Y.; Kamani, M. M.; and Mahdavi, M. 2020. Adaptive Personalized Federated Learning. *arXiv:2003.13461*.

Dvornik, N.; Schmid, C.; and Mairal, J. 2020. Selecting relevant features from a universal representation for few-shot classification. In *European Conference on Computer Vision*.

Fallah, A.; Mokhtari, A.; and Ozdaglar, A. 2020. Personalized Federated Learning with Theoretical Guarantees: A Model-Agnostic Meta-Learning Approach. In *Advances in Neural Information Processing Systems*.

Ghosh, A.; Chung, J.; Yin, D.; and Ramchandran, K. 2020. An Efficient Framework for Clustered Federated Learning. In *Advances in Neural Information Processing Systems*.

He, C.; Annavaram, M.; and Avestimehr, S. 2020. FedNAS: Federated Deep Learning via Neural Architecture Search. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

He, C.; Li, S.; So, J.; Zhang, M.; Wang, H.; Wang, X.; Vepakomma, P.; Singh, A.; Qiu, H.; Shen, L.; et al. 2020. Fedml: A research library and benchmark for federated machine learning. *arXiv:2007.13518*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hoang, M.; Hoang, N.; Low, B. K. H.; and Kingsford, C. 2019. Collective model fusion for multiple black-box experts. In *International Conference on Machine Learning*, 2742–2750. PMLR.

Hoang, N.; Lam, T.; Low, B. K. H.; and Jaillet, P. 2020. Learning Task-Agnostic Embedding of Multiple Black-Box Experts for Multi-Task Model Fusion. In *International Conference on Machine Learning*, 4282–4292. PMLR.

Jeong, E.; Oh, S.; Kim, H.; Park, J.; Bennis, M.; and Kim, S.-L. 2018. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-IID private data. In *Advances in Neural Information Processing Systems*.

Kairouz, P.; McMahan, H. B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A. N.; et al. 2019. Advances and open problems in federated learning. *arXiv:1912.04977*.

Krizhevsky, A. 2012. Learning Multiple Layers of Features from Tiny Images. *University of Toronto*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2017. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6): 84–90.

LeCun, Y. 1998. The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/. Accessed: 1998-11-01.

Li, D.; and Wang, J. 2020. Fedmd: Heterogenous federated learning via model distillation. In *Advances in Neural Information Processing Systems*.

Li, Q.; He, B.; and Song, D. 2021. Model-Contrastive Federated Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10713–10722.

Li, T.; Sahu, A. K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; and Smith, V. 2020. Federated optimization in heterogeneous networks. *MLSys*.

Liang, P. P.; Liu, T.; Ziyin, L.; Salakhutdinov, R.; and Morency, L.-P. 2020. Think Locally, Act Globally: Federated Learning with Local and Global Representations. *Advances in Neural Information Processing Systems*.

Lin, T.; Kong, L.; Stich, S. U.; and Jaggi, M. 2020. Ensemble Distillation for Robust Model Fusion in Federated Learning. In *Advances in Neural Information Processing Systems*.

Liu, L.; Hamilton, W. L.; Long, G.; Jiang, J.; and Larochelle, H. 2020. A Universal Representation Transformer Layer for Few-Shot Image Classification. In *International Conference on Learning Representations*.

Liu, Y.; Pan, S.; Jin, M.; Zhou, C.; Xia, F.; and Yu, P. S. 2021a. Graph self-supervised learning: A survey. *arXiv preprint arXiv:2103.00111*.

Liu, Y.; Pan, S.; Wang, Y. G.; Xiong, F.; Wang, L.; and Lee, V. 2021b. Anomaly Detection in Dynamic Graphs via Transformer. *IEEE Transactions on Knowledge and Data Engineering*.

Long, G.; Shen, T.; Tan, Y.; Gerrard, L.; Clarke, A.; and Jiang, J. 2021. Federated learning for privacy-preserving open innovation future on digital health. In *Humanity Driven AI*. Springer.

Long, G.; Tan, Y.; Jiang, J.; and Zhang, C. 2020. Federated Learning for Open Banking. In *Federated Learning*, 240–254. Springer.

Luo, J.; Wu, X.; Luo, Y.; Huang, A.; Huang, Y.; Liu, Y.; and Yang, Q. 2019. Real-world image datasets for federated learning. *arXiv:1910.11089*.

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

McMahan, H. B.; Moore, E.; Ramage, D.; et al. 2017. Communication-efficient learning of deep networks from decentralized data. *AISTATS*.

Michieli, U.; and Ozay, M. 2021. Prototype Guided Federated Learning of Visual Feature Representations. *arXiv preprint arXiv:2105.08982*.

Rieke, N.; Hancox, J.; Li, W.; Milletari, F.; Roth, H. R.; Albarqouni, S.; Bakas, S.; Galtier, M. N.; Landman, B. A.; Maier-Hein, K.; et al. 2020. The future of digital health with federated learning. *NPJ digital medicine*, 3(1): 1–7.

Sattler, F.; Müller, K.-R.; and Samek, W. 2020. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE transactions on neural networks and learning systems*.

Simonyan, K.; and Zisserman, A. 2014. Two-Stream Convolutional Networks for Action Recognition in Videos. In *Advances in Neural Information Processing Systems*, 568–576.

Singh, I.; Zhou, H.; Yang, K.; Ding, M.; Lin, B.; and Xie, P. 2020. Differentially-private federated neural architecture search. In *FL-International Conference on Machine Learning Workshop*.

Snell, J.; Swersky, K.; and Zemel, R. 2017. Prototypical Networks for Few-shot Learning. *Advances in Neural Information Processing Systems*, 30: 4077–4087.

Tan, A. Z.; Yu, H.; Cui, L.; and Yang, Q. 2021. Towards personalized federated learning. *arXiv preprint arXiv:2103.00710*.

Van der Maaten, L.; and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of machine learning research*, 9(11).

Wang, H.; Yurochkin, M.; Sun, Y.; Papailiopoulos, D.; and Khazaeni, Y. 2020a. Federated Learning with Matched Averaging. In *International Conference on Learning Representations*.

Wang, J.; Liu, Q.; Liang, H.; Joshi, G.; and Poor, H. V. 2020b. Tackling the Objective Inconsistency Problem in Heterogeneous Federated Optimization. *Advances in neural information processing systems*.

Wieting, J.; Bansal, M.; Gimpel, K.; and Livescu, K. 2015. Towards universal paraphrastic sentence embeddings. *arXiv:1511.08198*.

Xie, M.; Long, G.; Shen, T.; Wang, X.; Tianyi, Z.; and Jiang, J. 2020. Multi-center Federated Learning. *arXiv:2005.01026*.

Xu, J.; Glicksberg, B. S.; Su, C.; Walker, P.; Bian, J.; and Wang, F. 2020. Federated learning for healthcare informatics. *Journal of Healthcare Informatics Research*, 1–19.

Zhu, H.; Zhang, H.; and Jin, Y. 2020. From federated learning to federated neural architecture search: a survey. *Complex & Intelligent Systems*, 1–19.

Zhu, L.; Liu, Z.; and Han, S. 2019. Deep Leakage from Gradients. In *Advances in Neural Information Processing Systems*, 14774–14784.