Tackling Data Heterogeneity in Federated Learning with Class Prototypes

Yutong Dai[1], **Zeyuan Chen**[2], **Junnan Li**[2], **Shelby Heinecke**[2], **Lichao Sun**[1], **Ran Xu**[2]

[1] Lehigh University
[2] Salesforce Research
{yud319, lis221}@lehigh.edu, {zeyuan.chen, junnan.li, shelby.heinecke, ran.xu}@salesforce.com

Abstract

Data heterogeneity across clients in federated learning (FL) settings is a widely acknowledged challenge. In response, personalized federated learning (PFL) emerged as a framework to curate local models for clients' tasks. In PFL, a common strategy is to develop local and global models jointly - the global model (for generalization) informs the local models, and the local models (for personalization) are aggregated to update the global model. A key observation is that if we can improve the generalization ability of local models, then we can improve the generalization of global models, which in turn builds better personalized models. In this work, we consider class imbalance, an overlooked type of data heterogeneity, in the classification setting. We propose FedNH, a novel method that improves the local models' performance for both personalization and generalization by combining the uniformity and semantics of class prototypes. FedNH initially distributes class prototypes uniformly in the latent space and smoothly infuses the class semantics into class prototypes. We show that imposing uniformity helps to combat prototype collapse while infusing class semantics improves local models. Extensive experiments were conducted on popular classification datasets under the cross-device setting. Our results demonstrate the effectiveness and stability of our method over recent works.

Introduction

Federated learning (FL) (McMahan et al. 2017) is an emerging area that attracts significant interest in the machine learning community due to its capability to allow collaborative learning from decentralized data with privacy protection. However, in FL, clients may have different data distributions, which violates the standard independent and identically distribution (i.i.d) assumption in centralized machine learning. The non-i.i.d phenomenon is known as the data heterogeneity issue and is an acknowledged cause of the performance degradation of the global model (Tan et al. 2022a). Moreover, from the client's perspective, the global model may not be the best for their tasks. Therefore, personalized federated learning (PFL) emerged as a variant of FL, where personalized models are learned from a combination of the global model and local data to best suit client tasks.

While PFL methods address data heterogeneity, class imbalance combined with data heterogeneity remains overlooked. Class imbalance occurs when clients' data consists of different class distributions and the client may not possess samples of a particular class at all. Ideally, the personalized model can perform equally well in all classes that appeared in the local training dataset. For example, medical institutions have different distributions of medical records across diseases (Ng et al. 2021), and it is crucial that the personalized model can detect local diseases with equal precision. Meanwhile, the currently adopted practice of evaluating the effectiveness of PFL methods can also be biased. Specifically, when evaluating the accuracy, a single balanced testing dataset is split into multiple local testing datasets that match clients' training data distributions. Then each personalized model is tested on the local testing dataset, and the averaged accuracy is reported. However, in the presence of class imbalance, such an evaluation protocol will likely give a biased assessment due to the potential overfitting of the dominant classes. It is tempting to borrow techniques developed for centralized class imbalance learning, like re-sampling or re-weighting the minority classes. However, due to the data heterogeneity in the FL setting, different clients might have different dominant classes and even have different missing classes; hence the direct adoption may not be applicable. Furthermore, re-sampling would require the knowledge of all classes, potentially violating the privacy constraints.

Recent works in class imbalanced learning in non-FL settings (Kang et al. 2019; Zhou et al. 2020) suggest decoupling the training procedure into the representation learning and classification phases. The representation learning phase aims to build high-quality representations for classification, while the classification phase seeks to balance the decision boundaries among dominant classes and minority classes. Interestingly, FL works such as (Oh, Kim, and Yun 2021; Chen and Chao 2021) find that the classifier is the cause of performance drop and suggest that learning strong shared representations can boost performance.

Consistent with the findings in prior works, as later shown in Figure 1, we observe that representations for different classes are uniformly distributed over the representation space and cluster around the class prototype when learned with class-balanced datasets. However, when the training set is class-imbalanced, as is the case for different clients, rep-

resentations of minority classes overlap with those of majority classes; hence, the representations are of low quality. Motivated by these observations, we propose FedNH (nonparametric head), a novel method that imposes uniformity of the representation space and preserves class semantics to address data heterogeneity with imbalanced classes. We initially distribute class prototypes uniformly in the latent space as an inductive bias to improve the quality of learned representations and smoothly infuse the class semantics into class prototypes to improve the performance of classifiers on local tasks. Our contributions are summarized as follows.

- We propose FedNH, a novel method that tackles data heterogeneity with class imbalance by utilizing uniformity and semantics of class prototypes.
- We design a new metric to evaluate personalized model performance. This metric is less sensitive to class imbalance and reflects personalized model generalization ability on minority classes.
- Numerical experiments on Cifar10, Cifar100, and TinyImageNet show that FedNH can effectively improve both personalized and global model classification accuracy. The results are on par or better than the state-of-the-art, with significantly fewer computation costs (refer to the appendix for discussions on computation costs).

We close this section by introducing the notation and terminology used throughout the paper.

Notation and Terminology. Let $\mathbb{R}^n$ and $\mathbb{R}^{m\times n}$ denote the set of n dimensional real vectors and the set of m-by-n dimensional real matrices, respectively. Unless stated explicitly, $\|\cdot\|$ denotes the ℓ_2 norm and $|\cdot|$ denotes the set cardinality operator. Let $[N]$ denote the set $\{1, 2, \cdots, N\}$ for any positive integer N. For any matrix $A \in \mathbb{R}^{m\times n}$, A_i denotes the ith row of A. Let $\mathcal{N}_d(\mu, \sigma)$ denote the d-dimensional normal distribution with mean μ and variance σ^2. $X \sim \mathcal{N}_d(\mu, \sigma)$ represents a random sample from the distribution. $\mathbb{E}[\cdot]$ is the expectation operator and $\lceil\cdot\rceil$ is the round-up operator.

For a neural network, we decompose its parameters into the body (θ) and the head (W). The body is used to learn the abstract representation of inputs, and the head is used for classification. The output of a neural network can be written as $Wf(\theta; \cdot)$, where the second argument is a placeholder for an input. In this work, we solely consider the last linear layer as the head. We use the terms *head* and *prototype* interchangeably as they both refer to parameters of the last linear classification layer of the neural network. The term head is often used in discussing the neural network's architecture, while the term prototype appears more often when discussing the classification task. Some works define the prototype as the averaged representations of a class, and such a distinction will be made clear in the context.

Related Work

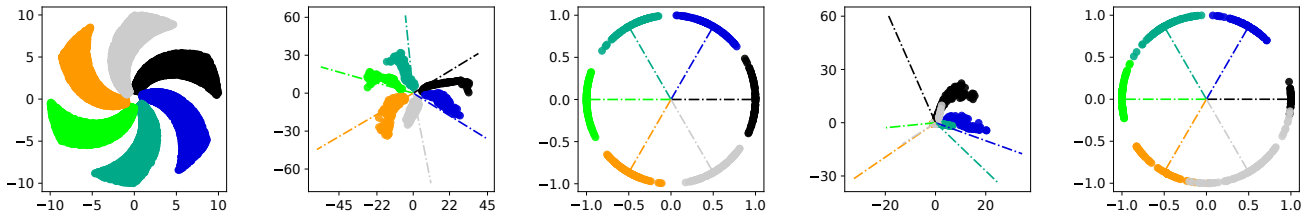
Personalized Federated Learning

PFL methods can be roughly classified based on the strategies to generate personalized models, e.g., parameter decoupling, regularization, and model interpolation. For a detailed discussion, we refer readers to the survey (Tan et al. 2022a). Here, we mainly focus on methods that decouple the body and head when clients perform local training. FedPer (Arivazhagan et al. 2019) learns the body and head concurrently as in FedAvg and only shares the body with the server. Therefore, a personalized model consists of a shared body and a personalized head. FedRep (Collins et al. 2021) learns the head and body sequentially and only shares the body. Specifically, each client first learns the head with a fixed body received from the server and then learns the body with the latest personalized head fixed. FedBABU (Oh, Kim, and Yun 2021) learns only the body with the randomly initialized and *fixed* head during local updates and shares only the body to the server. Personalized models are obtained by fine-tuning the global model when training is finished. FedROD (Chen and Chao 2021) designs a two-head-one-body architecture, where the two heads consist of a generalized head trained with class-balanced loss while the personalized head is trained with empirical loss. The body and generalized head are shared with the server for aggregation, and the personalized head is kept privately. The methods mentioned above assume that the model architectures are the same across clients. FedProto (Tan et al. 2022b) lifts such a restriction. It only shares the class prototypes (calculated as the averaged representations of each class) so that different clients can have different model architectures as bodies.

Class Imbalanced Learning

In the non-FL setting, data-level and algorithm-level approaches are the most common ways to address the class imbalance. Over-sampling minority classes or under-sampling majority classes are simple yet effective ways to create more balanced class distributions (Kubat and Matwin 1997; Chawla et al. 2002; He and Garcia 2009). However, they may be vulnerable to overfitting minority classes or information loss on majority classes. On the algorithm level, various class-balanced losses are proposed to assign different losses either sample-wise or class-wise (Lin et al. 2017; Khan et al. 2017; Cao et al. 2019; Cui et al. 2019). Recent works (Kang et al. 2019; Zhou et al. 2020) suggest decoupling the training procedure into the representation learning and classification phases, where strong representations are learned during the first phase while the classifier is re-balanced during the second phase. Wang et al. (2020) proposes a multi-expert framework, where each expert is only responsible for a subset of classes to minimize the bias on minority classes.

In the FL setting, a few works address the class imbalance. (Duan et al. 2020) selects clients with complimentary class distributions to perform updates, which requires clients to reveal the class distribution to the server. (Wang et al. 2021) assumes an auxiliary dataset with balanced classes is available on the server to infer the composition of training data and designs a ratio loss to mitigate the impact of imbalance. CReFF (Shang et al. 2022) extends the idea from (Kang et al. 2019) to re-train a classifier based on federated features in a privacy preserving manner. To our best knowledge, FedROD and CReFF are the only works that address the data heterogeneity with the class imbalance in the FL setting without potential privacy leakage and without requiring auxiliary datasets on the server side.



(a) Spiral Dataset (b) Balanced + FH (c) Balanced + UH (d) Imbalanced + FH (e) Imbalanced + UH

Figure 1: Visualization of the learned representations and class prototypes. One color represents one class. (a): Balanced training dataset. (b): Representations (dots) and class prototypes (dashed lines) learned under the balanced dataset with the free head (FH), i.e, the classification head is also learnable. (c): Representations and class prototypes learned under the balanced dataset with the uniform head (UH). (d) and (e) are parallel to (c) and (e), respectively, with the imbalanced training dataset.

Methodology

A Motivating Example

In PFL, a crucial part of training is done locally in each client. We motivate our method by considering the behavior of learned representations of inputs and class prototypes learned by a neural network with the cross-entropy loss under both balanced and imbalanced training datasets. We first generate a balanced two-dimensional synthetic spiral training dataset (Figure 1(a)) with six classes, each with 3000 points. For each class $k \in \{0, \cdots, 5\}$, the data points are generated as $\mathcal{C}_k = \{(x_{k,i}, y_{k,i}) \mid x_{k,i} = r_i \sin \omega_{k,i}, y_{k,i} = r_i \cos \omega_{k,i}, i \in [3000]\}$, where for all $i \in [3000]$,

$$r_i = 1+(i-1)\frac{9}{2999} \text{ and } \omega_i = \frac{k}{3}\pi+(i-1)\frac{k}{3\times 2999}\pi+b_i,$$

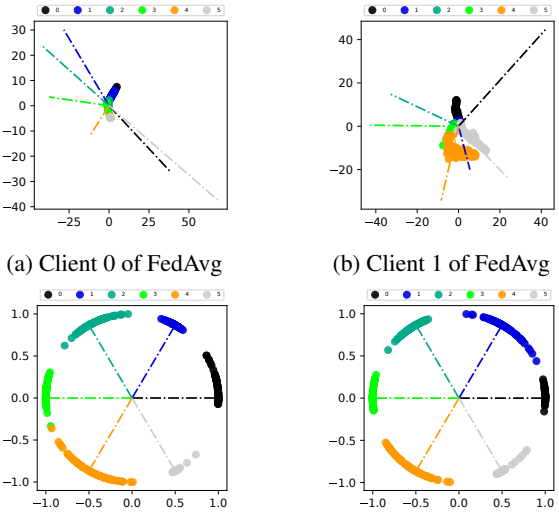
and $b_i \sim \mathcal{N}_1(0,1)$. A balanced testing dataset is created similarly (shown in the appendix). We consider a 5-layer fully connected neural network with ReLu activation (Agarap 2018) as the classifier. For visualization purposes, the dimension of the latent representation is set to two. We draw latent representations (points) and class prototypes (dashed lines)[1] over the testing dataset in Figure 1(b). Two key observations include that 1) the norm of class prototypes are of about the same magnitude and 2) class prototypes distribute relatively uniformly over the latent space. Next, we generate an imbalanced training dataset, where the number of samples for each class is selected sequentially from $\{3000, 1500, 750, 375, 187, 93\}$. Then as shown in Figure 1(d), the learned class prototypes are not of the same magnitude, the minority classes' prototypes are pulled towards the majority classes' prototypes, and the representations of minority classes collapse into the space that the majority ones occupy. This phenomenon matches with the previous study (Kang et al. 2019) and motivates us to distribute class prototypes uniformly over a unit hypersphere and fix prototypes throughout the training phase. As shown in Figure 1(c), enforcing class prototypes' uniformity will not hurt the training when the training dataset is balanced. And when the training set is imbalanced, from Figure 1(e), one can see the uniformity of class prototypes is an effective inductive bias to combat the negative impact of the dominating classes over the minority classes. We end this section by making the following remarks.

- Enforcing uniformity over the class prototypes can be regarded as a method of reweighing the training samples. For the samples that cannot be easily and correctly classified, they make more contributions to the cross-entropy loss. This uniformity strategy differs from methods such as Focal loss (Lin et al. 2017), which add explicit weights per sample. Meanwhile, this strategy does not require tuning any parameters. Moreover, it brings additional benefits to the minority classes, whose representations are no longer overlapped with the majority classes.
- Fixing class prototypes sets a consistent learning goal for clients (Oh, Kim, and Yun 2021) and imposing uniformity over class prototypes helps to combat minority representation collapse in the FL setting. We empirically validate this claim by distributing the spiral training set to 100 clients and performing learning with FedAvg. More details about the experiment can be found in the appendix. Figure 2(a) and Figure 2(b) visualize class prototypes and latent feature representations for two different clients learned with FedAvg. Since prototypes are free to move, both prototypes and representations for the same class occupy different places in different clients. Fixing prototypes with a uniformity constraint addresses such an issue as shown in Figure 2(c) and Figure 2(d).
- Independently, a related idea is also explored in non-FL setting (Li et al. 2022). However, our method is different from this work since our class prototypes take into account the class semantic similarity while class prototypes are fixed and never get updated in (Li et al. 2022).

Proposed Method

As we demonstrated in the motivating example, class prototypes learned from balanced datasets are distributed uniformly in the latent space. Hence, encouraging class prototypes learned from unbalanced data to also be distributed uniformly may be a way to improve performance. We also note that since the motivating example was produced with synthetic data, class semantics were not discussed. In practice, some classes are more semantically similar to others.

[1]We recall that the weight matrix of the last linear layer represents the prototypes, and each row of the matrix is also a vector of the same dimension as the latent representation.



(a) Client 0 of FedAvg (b) Client 1 of FedAvg

(c) Client 0 of FedAvg+UH (d) Client 1 of FedAvg+UH

Figure 2: Learned representations are consistent for different clients for FedAvg with the uniform classification head compared with vanilla FedAvg.

For example, the (wolf, dog) pair is more similar than the (wolf, plant) pair. Capturing semantics helps with performance (Su and Jurie 2012). This leads to two questions:

- (1) Given imbalanced data, how can we generate class prototypes such that pairs of prototypes are maximally separated and equidistant?
- (2) How can we encourage class prototypes to learn class semantics?

We address the first question by proposing a parameter initialization scheme for the classification head during FL training. We answer the second question by proposing an approach to infuse class semantics throughout the training process. Both answers are described in more detail in the following sections.

Initialization. The server is responsible for initializing parameters for both the body and classification head. The parameters for the body are initialized using common techniques, like Kaiming initialization (He et al. 2015) and Xavier initialization (Glorot and Bengio 2010). However, we initialize the parameter W for the head in a way to address the first question. Specifically, assuming there are $|\mathcal{C}|$ classes and the latent representation lives in a d-dimensional space, then one can solve a constrained optimization problem (1).

$$\max_{\{w_1,\cdots,w_{|\mathcal{C}|},M\}} M^2 \quad (1)$$
$$\text{s.t. } \|w_i - w_j\|^2 \geq M^2, \|w_i\|^2 = 1 \text{ for all } i \in [|\mathcal{C}|], i \neq j.$$

The first constraint is imposed to satisfy the maximally separated and equidistant requirements, while the second constraint is added to ensure all class prototypes are of the same magnitude. The problem can be solved by, for example, the interior point method (Wright, Nocedal et al. 1999). We remark that (Li et al. 2022) uses a different approach to generate class prototypes, which does not ensure that each prototype is of the same magnitude. When d and $|\mathcal{C}|$ are large, the computation cost might be high. However, the computation only needs to be performed once at the initialization stage and saved for future uses. Lastly, we remark that the body of the network internally adds a normalization layer to its output such that $\|f(\theta; x)\| = 1$ for any θ and valid input x. This normalization technique adds no extra parameters and negligible computation cost. It is also used in face verification tasks (Wang et al. 2017; Cheng et al. 2018).

Client Update. For the tth communication round, the kth client receives the body parameter θ^t and the head parameter W^t. Together with its local training dataset $\mathcal{D}_k$, it aims to achieve two goals. The first goal is to learn a strong body that can extract representations used for classification. To this goal, the kth client approximately solves the problem (2).

$$\theta_k^{t+1} \approx \arg\min_{\theta} \Big[F(\theta; \mathcal{D}_k, W^t, s)$$
$$:= \frac{1}{|\mathcal{D}_k|} \sum_{(x_i,y_i)\in\mathcal{D}_k} -\log \frac{\exp(s \cdot W_{y_i}^t f(\theta; x_i))}{\sum_{j=1}^{|\mathcal{C}|} \exp(s \cdot W_j^t f(\theta; x_i))}\Big], \quad (2)$$

where s is a fixed constant scalar. The s is added to compensate for the fact that $\left\|W_{y_i}^t f(\theta; x_i))\right\| \leq 1$ [2]. Specifically, the loss $F(\theta; \mathcal{D}_k, W^t, s)$ is lower bounded by a quantity inversely proportional to $s \cdot \left\|W_{y_i}^t f(\theta; x_i))\right\|$ (Wang et al. 2017, Proposition 2). If $s = 1$, it hinders the optimization process of learning a strong body. In practice, s can be set to some constants like 30 as suggested in (Cheng et al. 2018). The inverse of s is also called temperature in knowledge distillation (Hinton et al. 2015).

To approximately solve problem (2), we assume J steps of stochastic gradient descent are performed, i.e.,

$$\theta_k^{t,j} = \theta_k^{t,j-1} - \eta_t G_k(\theta_k^{t,j-1}; W^t) \text{ for } j \in [J],$$

where $\theta_k^{t,j}$ is the body parameter at the kth client in tth communication round after the jth local update and $G_k(\theta_k^{t,j-1}; W^t)$ is the stochastic gradient estimator (See Assumption 2). It is clear that $\theta_k^{t,J} = \theta_k^t$ and $\theta_k^{t,0} = \theta^{t-1}$. It is worth mentioning that the classification head is fixed during the local training so that all selected clients have a consistent learning goal under the guidance of the same head W^t.

Once the local training is done, the client completes its second goal, i.e., computing the averaged representations per class contained in $\mathcal{C}_k$ as (3)

$$\mu_{k,c}^{t+1} = \begin{cases} \frac{1}{|\{i:y_i=c\}|} \sum_{\{i:y_i=c\}} r_{k,i}^{t+1}, & \text{if } c \in \mathcal{C}_k. \\ 0, & \text{o.w. }, \end{cases} \quad (3)$$

where $\mathcal{C}_k$ is the set of classes that the kth client owns and $r_{k,i}^{t+1} = f(\theta_k^{t+1}; x_i, y_i)$ for $(x_i, y_i) \in \mathcal{D}_k$. These averaged representations $\{\mu_{k,c}^{t+1}\}_{c\in\mathcal{C}_k}$ provide a succinct local description of class prototypes for classes from $\mathcal{C}_k$. Finally, both the body parameters θ_k^{t+1} and local prototypes $\{\mu_{k,c}^{t+1}\}_{c\in\mathcal{C}_k}$ are sent back to the server for aggregation.

[2] Both the latent representation $f(\theta; x_i)$ and the prototype $W_{y_i}^t$ are normalized.

Algorithm 1: FedNH

Require: Total number of clients K; participation rate γ; number of communication rounds R; the smoothing parameter $\rho \in (0, 1)$; a set of full classes $\mathcal{C}$; the set of classes that the kth client owns as $\mathcal{C}_k$.

Initialization: Set the class prototype $W \in \mathbb{R}^{|\mathcal{C}| \times d}$ with W^0 and the θ with θ^0.

for $t = 0, \ldots, R-1$ communication rounds **do**
  Sample a subset $\mathcal{S}^t$ of clients with $|\mathcal{S}^t| = \lceil \gamma K \rceil$.
  Broadcast $\{\theta^t, W^t\}$ to clients $k \in \mathcal{S}^t$.
  for each client $k \in \mathcal{S}^t$ **in parallel do**
    $\left(\theta_k^{t+1}, \mu_k^{t+1}\right) \leftarrow \text{ClientUpdate}(\theta^t, W^t)$.
  end for
  Perform the global class prototype update as (4).
  Normalize W_c to have unit norm for all $c \in \mathcal{C}$.
  Perform aggregation as $\theta^{t+1} = \frac{1}{|\mathcal{S}^t|} \sum_{k \in \mathcal{S}^t} \theta_k^t$.
end for
procedure CLIENTUPDATE(θ^t, W^t)
  Initialize the local representation network with θ^t and class prototypes with W^t.
  Approximately solve the problem (2) to obtain θ_k^{t+1}.
  Perform the local class prototypes update as (3).
  Return $\left(\theta_k^{t+1}, \mu_k^{t+1}\right)$.
end procedure

Server Update. For the tth communication round, assume that clients from a set $\mathcal{S}^t$ respond to the server. Then the server aggregates the body by taking the average of the received $\{\theta_k^{t+1}\}_{k \in \mathcal{S}^t}$, which is the same as in FedAvg. However, for the global prototype update, we propose a new strategy that infuses class semantics, addressing the second question. Specifically, for $c \in \mathcal{C}$,

$$W_c^{t+1} \leftarrow \rho W_c^t + (1-\rho) \sum_{k \in S^t} \alpha_k^{t+1} \mu_{k,c}^{t+1}, \tag{4}$$

where the aggregation weights $\{\alpha_k^{t+1}\}_{k \in \mathcal{S}^t}$ are some positive scalars and with smoothing parameter $\rho \in (0, 1)$. A specific choice of these parameters is discussed in the appendix. We remark that ρ is suggested to set close to 1 for two reasons. First, at the initial training stage, the body of a network is not well trained; hence the learned representations and the clients' proposed prototypes are less informative. In this stage, we utilize the uniformity prototypes initialized at the server to guide the local training. Second, due to the client sampling, it is ideal that prototypes on the server side change gradually, hence using the $1-\rho$ to discount the contribution of local clients to the update of prototypes on the server side. The importance of ρ will also be justified by the convergence analysis in Section . It is worth mentioning class prototypes are updated in a non-parametric manner instead of by gradients. We end this section by formally summarizing our method FedNH in Algorithm 1.

Convergence Analysis

To avoid cluttered notations, we denote the local loss $F_k(\theta; W) \equiv F(\theta; \mathcal{D}_k, W, s)$. We first present a set of assumptions required to perform the convergence analysis.

Assumption 1. For all $k \in [K]$, $F_k(\theta; W)$ is bounded below, and $F_k(\theta; W)$ is L_g smooth with respect to its first argument, i.e., for any fixed $W \in \mathbb{R}^{d_2}$ and for all $(\theta_1, \theta_2) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_1}$.

$$\|\nabla_\theta F_k(\theta_1; W) - \nabla_\theta F_k(\theta_2; W)\| \leq L_g \|\theta_1 - \theta_2\|.$$

Assumption 2. Denote $G_k(\theta; W)$ as the stochastic gradient, where only a random subset of $\mathcal{D}_k$ is used to estimate the true partial gradient $\nabla_\theta F_k(\theta; W)$. There is a constant $\sigma > 0$, such that for all $k \in [K]$ and $(\theta, W) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2}$,

$$\mathbb{E}[G_k(\theta; W)] = \nabla_\theta F_k(\theta; W), \text{ and}$$
$$\mathbb{E}[\|G_k(\theta; W) - \nabla_\theta F_k(\theta; W)\|^2] \leq \sigma^2.$$

Assumption 3. Assume $\mathbb{E}\left[\|G(\theta; W)\|\right]^2 \leq M_G^2$ and $\|\nabla_\theta f(\theta; x, y)\| \leq M_f$ for all $(\theta, W) \in \mathcal{B}$ and $(x, y) \in \bigcup_k \mathcal{D}_k$, where $\mathcal{B}$ is a bounded set.

Assumption 1 is parallel to the common smoothness assumption made in the literature(Li et al. 2020; Karimireddy et al. 2020; Acar et al. 2020), where we slightly relax it by only requiring F_k to be smooth with respect to its first argument. Assumption 2 is also a standard assumption on the first- and second-order momentum of the stochastic gradient estimate $G_k(\theta; W)$. Assumption 3 imposes a bound on the gradient, which is reasonable as long as (θ, W) remains bounded and it is also assumed in (Stich 2019; Yu, Yang, and Zhu 2019; Tan et al. 2022b).

We aim to establish the convergence results for each client. The challenge in the analysis is that, for each round t, the objective function $F_k(\theta; W^t)$ is different as W^t changes. Thus, the type of convergence result we can expect is that for some $(\theta_k^{t,j}, W^t)$, the $\left\|\nabla_\theta F_k(\theta^{t,j}; W^t)\right\|$ is small. An informal statement of the convergence result is given below. The formal statement and proofs can be found in the appendix.

Theorem 1 (Informal)**.** *Let the kth client uniformly at random returns an element from $\{\theta_k^{t,j}\}$ as the solution, denoted as θ_k^*. Further, let W^* share the same round index as θ_k^*. Then for any $\epsilon > 0$, set $\rho \in (\nu_1(\epsilon, M_G, M_f), 1)$ and $\eta \in \left(0, \nu_2(\epsilon, L_g, \sigma^2, \rho, M_G, M_f)\right)$, if $R > \mathcal{O}(\epsilon^{-1})$, one gets*

$$\mathbb{E}\left[\left\|\nabla_\theta F_k(\theta_k^*; W^*)\right\|^2\right] \leq \epsilon,$$

where $\nu_1(\epsilon, M_G, M_f)$, $\nu_2(\epsilon, L_g, \sigma^2, \rho, M_G, M_f)$, M_G, and M_f are some positive constants.

Experiments

Setups, Evaluation, and Baselines

Setups. Extensive experiments are conducted on Cifar10, Cifar100 (Krizhevsky 2009), and TinyImageNet three popular benchmark datasets. We follow (Chen and Chao 2021) to use a simple Convolutional neural network for both Cifar10 and Cifar100 datasets while we use Resnet18 (He et al. 2016) for the TinyImageNet. To simulate the heterogeneity with class imbalance, we follow (Lin et al. 2020) to distribute each class to clients using the Dirichlet(β) distribution with $\beta \in \{0.3, 1.0\}$, resulting in clients having different class distributions and a different number of

Dataset	Method	Dir(0.3)			Dir(1.0)		
		GM	PM(V)	PM(L)	GM	PM(V)	PM(L)
Cifar10	Local	—	42.79 ± 2.45	71.57 ± 1.82	—	41.20 ± 1.27	58.34 ± 1.03
	FedAvg	66.40 ± 3.13	63.10 ± 1.33	84.08 ± 2.31	73.07 ± 1.60	68.07 ± 1.23	79.12 ± 2.11
	FedPer	61.58 ± 0.43	59.66 ± 2.34	82.38 ± 1.50	63.33 ± 0.53	60.66 ± 2.17	73.40 ± 1.36
	Ditto	66.40 ± 3.13	53.69 ± 1.11	80.08 ± 2.17	73.07 ± 1.60	61.22 ± 1.77	74.78 ± 2.16
	FedRep	40.13 ± 0.17	56.47 ± 2.31	80.22 ± 2.45	47.92 ± 0.38	55.06 ± 2.27	68.99 ± 1.27
	FedProto	—	41.48 ± 1.02	68.35 ± 1.75	—	39.65 ± 1.33	53.23 ± 1.78
	CReFF	66.46 ± 1.40	63.10 ± 2.16	84.08 ± 2.31	71.63 ± 0.61	68.07 ± 1.44	79.12 ± 2.11
	FedBABU	62.78 ± 3.09	60.58 ± 2.16	82.64 ± 1.03	70.34 ± 1.72	65.49 ± 1.44	77.35 ± 1.80
	FedROD	**72.31 ± 0.16**	**65.66 ± 1.27**	83.44 ± 1.03	**75.50 ± 0.15**	69.18 ± 1.98	77.84 ± 1.76
	FedNH	69.01 ± 2.51	65.02 ± 1.23	**84.63 ± 2.11**	75.34 ± 0.86	**69.64 ± 1.15**	**79.53 ± 2.14**
Cifar100	Local	—	13.63 ± 2.45	30.89 ± 1.82	—	9.44 ± 1.27	16.71 ± 1.03
	FedAvg	35.14 ± 0.48	31.85 ± 1.33	50.77 ± 2.31	36.07 ± 0.41	28.86 ± 1.23	38.35 ± 2.11
	FedPer	15.04 ± 0.06	16.15 ± 2.34	33.10 ± 1.50	14.69 ± 0.03	11.61 ± 2.17	19.08 ± 1.36
	Ditto	35.14 ± 0.48	26.19 ± 1.11	45.91 ± 2.17	36.07 ± 0.41	22.92 ± 1.77	32.81 ± 2.16
	FedRep	5.42 ± 0.03	13.59 ± 2.31	29.45 ± 2.45	6.37 ± 0.04	9.47 ± 2.27	16.07 ± 1.27
	FedProto	—	10.64 ± 1.02	19.11 ± 1.75	—	9.24 ± 1.33	12.61 ± 1.78
	CReFF	22.90 ± 0.30	31.85 ± 1.33	50.77 ± 2.31	22.21 ± 0.15	28.86 ± 1.23	38.35 ± 2.11
	FedBABU	32.41 ± 0.40	28.96 ± 2.16	47.86 ± 1.03	32.34 ± 0.49	25.84 ± 1.44	34.85 ± 1.80
	FedROD	33.83 ± 0.25	28.53 ± 1.27	42.93 ± 1.03	35.20 ± 0.19	27.58 ± 1.98	33.44 ± 1.76
	FedNH	**41.34 ± 0.25**	**38.25 ± 1.23**	**55.21 ± 2.11**	**43.19 ± 0.24**	**36.88 ± 1.15**	**45.46 ± 2.14**
TinyImageNet	Local	—	7.55 ± 2.45	19.94 ± 1.82	—	5.10 ± 1.27	9.93 ± 1.03
	FedAvg	34.63 ± 0.26	27.35 ± 1.33	44.97 ± 2.31	37.65 ± 0.37	28.82 ± 1.23	37.15 ± 2.11
	FedPer	15.28 ± 0.14	13.84 ± 2.34	30.72 ± 1.50	13.71 ± 0.07	9.82 ± 2.17	17.05 ± 1.36
	Ditto	34.63 ± 0.26	23.85 ± 1.11	42.67 ± 2.17	37.65 ± 0.37	24.97 ± 1.77	34.70 ± 2.16
	FedRep	3.27 ± 0.02	9.24 ± 2.31	22.86 ± 2.45	3.91 ± 0.03	5.76 ± 2.27	10.86 ± 1.27
	FedProto	—	5.17 ± 1.02	10.44 ± 1.75	—	4.21 ± 1.33	6.34 ± 1.78
	CReFF	25.82 ± 0.41	27.35 ± 1.33	44.97 ± 2.31	27.87 ± 0.38	28.82 ± 1.23	37.15 ± 2.11
	FedBABU	26.36 ± 0.32	20.85 ± 2.16	37.96 ± 1.03	30.25 ± 0.32	22.74 ± 1.44	31.01 ± 1.80
	FedROD	36.46 ± 0.28	28.23 ± 1.27	45.26 ± 1.03	37.71 ± 0.31	29.65 ± 1.98	**38.43 ± 1.76**
	FedNH	**36.71 ± 0.36**	**30.99 ± 1.23**	**46.14 ± 2.11**	**38.68 ± 0.30**	**30.58 ± 1.15**	38.25 ± 2.14

Table 1: Comparison of testing accuracy. The best results are in bold font while the second best results are underlined. The lines "—" represent results are not available. The numbers (mean ± std) are the average of three independent runs.

samples. Note that when $\beta \leq 1.0$, each client is likely to have one or two dominating classes while owning a few or even zero samples from the remaining classes. Consequently, both classes and the number of samples are imbalanced among clients. To simulate the cross-device setting, we consider 100 clients with a 10% participation ratio.

Evaluation Metrics. We evaluate the testing accuracy of both global and personalized models. We take the latest local models as the personalized model for methods that do not explicitly produce personalized models. For PFL methods that do not upload the head to the server, we use the head initialized at the starting point. We follow (Chen and Chao 2021) to evaluate personalized models on a single testing *balanced* dataset $\mathcal{D}^{\text{test}}$ to reduce the randomness from dataset partitioning. Specifically, the accuracy of the ith personalized model is computed as

$$\text{acc}_i = \frac{\sum_{(x_j,y_j)\sim\mathcal{D}^{\text{test}}} \alpha_i(y_j)\,\mathbf{1}(y_j = \hat{y}_j)}{\sum_{(x_j,y_j)\sim\mathcal{D}^{\text{test}}} \alpha_i(y_j)},$$

where $\alpha_i(\cdot)$ is a positive-valued function and $\mathbf{1}(\cdot)$ is the indicator function. y_j and $\hat{y}_j$ are the true label and predicted label, respectively. We consider two choices of setting $\alpha_i(\cdot)$. The first choice is to set $\alpha_i(y) = \mathbb{P}_i(y = c)$, where $\mathbb{P}_i(y = c)$ stands for the probability that the sample y is from class c in the ith client. The probability $\mathbb{P}_i(y = c)$ can be estimated from the ith client's training dataset. The second choice sets $\alpha_i(y)$ to 1 if the class y appears in ith client's training dataset and 0 otherwise. This metric measures the generalization ability of personalized models because it treats all classes presented in the local training dataset with equal importance.

Baselines. We choose several popular and state-of-the-art FL/PFL algorithms, such as FedAvg (McMahan et al. 2017), FedPer (Arivazhagan et al. 2019), Ditto (Li et al. 2021), FedRep (Collins et al. 2021), FedProto (Tan et al. 2022b), FedBABU (Oh, Kim, and Yun 2021), FedROD (Chen and Chao 2021), and CReFF (Shang et al. 2022).

Experimental environments and implementation details on all chosen methods are deferred to the appendix.

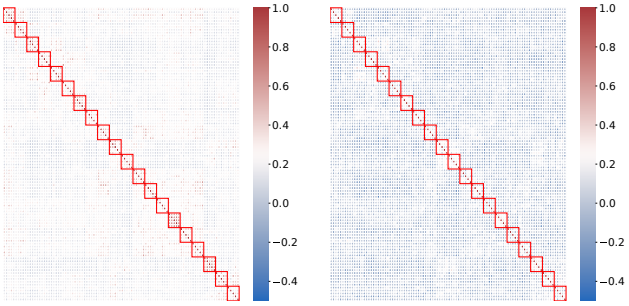


Figure 3: Similarity of prototypes learned with FedNH (left) and FedAvg (right). x and y axes represent class indices.

Results

We report three types of testing accuracy in Table 1. Numbers under the GM column are the averaged testing accuracy for the global model (mean±standard deviation); PM(V) reports the averaged testing accuracy of personalized models by assigning equal weights to all appearing classes (the second choice of setting the weight function $\alpha_i(\cdot)$); PM(L) represents the averaged testing accuracy by assuming the training and testing dataset has the same distribution (the first choice of setting the weight function $\alpha_i(\cdot)$). Several observations are made:

- Personalized models from FedAvg serve as the strong baseline (evaluated under both PM(L) and PM(v) metrics) and can outperform many specialized personalized methods. Similar observations are also made in (Oh, Kim, and Yun 2021; Chen and Chao 2021).
- In the cross-device setting, the performance gain of the personalized models from FedNH is the most consistent across all tested cases. Further, FedNH has the best PM(V) testing accuracy in almost all cases. Moreover, it adds minimum computation overhead compared with state-of-the-art methods like FedROD.
- The PM(V) metric is less sensitive to class imbalance than PM(L). Note that the accuracy of PM(L) changes significantly from Dir(0.3) to Dir(1.0), while PM(L) is relatively stable, as shown in Table 1.
- A strong global model often leads to strong personalized models by comparing the GM and PM metrics.

Discussions on why some methods like Fedproto and CReFF do not perform well are discussed in the appendix.

Analysis

Capture of Class Semantics. To further validate that prototypes learned by FedNH can capture the class semantics, we visualize the pairwise class prototypes' similarity from Cifar100 in Figure 3. 100 classes in Cifar100 form 20 super-classes. In Figure 3, we group classes from one super-class into one red box along the main diagonal. Because the classes within one red box are semantically closer, their prototypes should also be more similar. We can see that our FedNH learns the semantics of classes by capturing their fine-grained similarities while the FedAvg simply treats all classes as different. Similar plots for other methods are given in appendix with a more detailed analysis. We also visualize learned representations (in the appendix) for different classes on different clients and find that similar classes' representations tend to be closer.

Sensitivity Analysis. We conduct the sensitivity analysis on the smoothing parameter ρ by choosing it from $\{0.1, 0.3, 0.5, 0.7, 0.9\}$ and plotting the relative performance gain over the base case $\rho = 0.1$. As shown in Figure 4, $\rho = 0.1$ gives the worst performance. This matches Theorem 1 that suggests ρ cannot be too small.

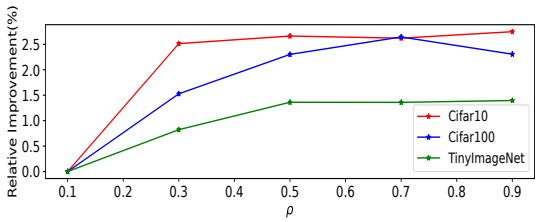


Figure 4: Sensitivity analysis on the smoothing parameter ρ.

Fairness Analysis. We measure fairness by computing the standard deviation of the accuracy of personalized models across all clients. A smaller standard deviation indicates that all clients' performance tends to be concentrated around the mean accuracy. In Table 2, we present the result in the form of mean accuracy ± standard deviation. One can see that FedNH improves all clients more equally.

	Cifar10	Cifar100	TinyImageNet
Local	71.57 ± 10.13	30.89 ± 4.60	19.94 ± 3.27
FedAvg	84.08 ± 6.80	50.77 ± 4.23	44.97 ± 2.99
FedPer	82.38 ± 6.38	33.10 ± 4.26	30.72 ± 3.21
Ditto	80.08 ± 7.83	45.91 ± 4.10	42.67 ± 3.05
FedRep	80.22 ± 7.06	29.45 ± 4.19	22.86 ± 3.20
FedProto	68.35 ± 11.03	19.11 ± 4.17	10.44 ± 2.80
FedBABU	82.64 ± 6.11	47.86 ± 3.89	37.96 ± 2.79
FedROD	83.44 ± 5.89	42.93 ± 4.10	45.26 ± 2.72
FedNH	84.63 ± **5.61**	55.21 ± **3.91**	46.14 ± **2.70**

Table 2: Compare the fairness of baseline methods.

Conclusion

In this work, we proposed FedNH, a novel FL algorithm to address the data heterogeneity with class imbalance. FedNH combines the uniformity and semantics of class prototypes to learn high-quality representations for classification. Extensive experiments were conducted to show the effectiveness and robustness of our method over recent works.

Limitation and Future Work. Our idea currently only applies to the classification task, and the inductive bias from uniformity and semantics of class prototypes can only be imposed on the head of neural network architecture. Our future work will explore the possibility of extending the inductive bias to the intermediate layers of a neural network and different vision tasks.

References

Acar, D. A. E.; Zhao, Y.; Matas, R.; Mattina, M.; Whatmough, P.; and Saligrama, V. 2020. Federated Learning Based on Dynamic Regularization. In *International Conference on Learning Representations*.

Agarap, A. F. 2018. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*.

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

Cao, K.; Wei, C.; Gaidon, A.; Arechiga, N.; and Ma, T. 2019. Learning imbalanced datasets with label-distribution-aware margin loss. *Advances in neural information processing systems*, 32.

Chawla, N. V.; Bowyer, K. W.; Hall, L. O.; and Kegelmeyer, W. P. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16: 321–357.

Chen, H.-Y.; and Chao, W.-L. 2021. On Bridging Generic and Personalized Federated Learning for Image Classification. In *International Conference on Learning Representations*.

Cheng, J.; Wang, F.; Liu, W.; and Liu, H. 2018. Additive margin softmax for face verification. *IEEE Signal Processing Letters*, 25(7): 926–930.

Collins, L.; Hassani, H.; Mokhtari, A.; and Shakkottai, S. 2021. Exploiting shared representations for personalized federated learning. In *International Conference on Machine Learning*, 2089–2099. PMLR.

Cui, Y.; Jia, M.; Lin, T.-Y.; Song, Y.; and Belongie, S. 2019. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 9268–9277.

Duan, M.; Liu, D.; Chen, X.; Liu, R.; Tan, Y.; and Liang, L. 2020. Self-balancing federated learning with global imbalanced data in mobile systems. *IEEE Transactions on Parallel and Distributed Systems*, 32(1): 59–71.

Glorot, X.; and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 249–256. JMLR Workshop and Conference Proceedings.

He, H.; and Garcia, E. A. 2009. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9): 1263–1284.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, 1026–1034.

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.

Hinton, G.; Vinyals, O.; Dean, J.; et al. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7).

Kang, B.; Xie, S.; Rohrbach, M.; Yan, Z.; Gordo, A.; Feng, J.; and Kalantidis, Y. 2019. Decoupling Representation and Classifier for Long-Tailed Recognition. In *International Conference on Learning Representations*.

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

Khan, S. H.; Hayat, M.; Bennamoun, M.; Sohel, F. A.; and Togneri, R. 2017. Cost-sensitive learning of deep feature representations from imbalanced data. *IEEE transactions on neural networks and learning systems*, 29(8): 3573–3587.

Krizhevsky, A. 2009. Learning Multiple Layers of Features from Tiny Images. *MSc thesis*.

Kubat, M.; and Matwin, S. 1997. Addressing the Curse of Imbalanced Training Sets: One-Sided Selection. In *ICML*.

Li, T.; Cao, P.; Yuan, Y.; Fan, L.; Yang, Y.; Feris, R. S.; Indyk, P.; and Katabi, D. 2022. Targeted supervised contrastive learning for long-tailed recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6918–6928.

Li, T.; Hu, S.; Beirami, A.; and Smith, V. 2021. Ditto: Fair and robust federated learning through personalization. In *International Conference on Machine Learning*, 6357–6368. PMLR.

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.

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.

Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; and Dollár, P. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, 2980–2988.

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*, 1273–1282. PMLR.

Ng, D.; Lan, X.; Yao, M. M.-S.; Chan, W. P.; and Feng, M. 2021. Federated learning: a collaborative effort to achieve better medical imaging models for individual sites that have small labelled datasets. *Quantitative Imaging in Medicine and Surgery*, 11(2): 852.

Oh, J.; Kim, S.; and Yun, S.-Y. 2021. FedBABU: Toward Enhanced Representation for Federated Image Classification. In *International Conference on Learning Representations*.

Shang, X.; Lu, Y.; Huang, G.; and Wang, H. 2022. Federated Learning on Heterogeneous and Long-Tailed Data via Classifier Re-Training with Federated Features. *arXiv preprint arXiv:2204.13399*.

Stich, S. U. 2019. Local SGD Converges Fast and Communicates Little. In *ICLR 2019-International Conference on Learning Representations*, CONF.

Su, Y.; and Jurie, F. 2012. Improving image classification using semantic attributes. *International journal of computer vision*, 100(1): 59–77.

Tan, A. Z.; Yu, H.; Cui, L.; and Yang, Q. 2022a. Towards personalized federated learning. *IEEE Transactions on Neural Networks and Learning Systems*.

Tan, Y.; Long, G.; Liu, L.; Zhou, T.; Lu, Q.; Jiang, J.; and Zhang, C. 2022b. Fedproto: Federated prototype learning across heterogeneous clients. In *AAAI Conference on Artificial Intelligence*, volume 1, 3.

Wang, F.; Xiang, X.; Cheng, J.; and Yuille, A. L. 2017. Normface: L2 hypersphere embedding for face verification. In *Proceedings of the 25th ACM international conference on Multimedia*, 1041–1049.

Wang, L.; Xu, S.; Wang, X.; and Zhu, Q. 2021. Addressing class imbalance in federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 10165–10173.

Wang, X.; Lian, L.; Miao, Z.; Liu, Z.; and Yu, S. 2020. Long-tailed Recognition by Routing Diverse Distribution-Aware Experts. In *International Conference on Learning Representations*.

Wright, S.; Nocedal, J.; et al. 1999. Numerical optimization. *Springer Science*, 35(67-68): 7.

Yu, H.; Yang, S.; and Zhu, S. 2019. Parallel restarted SGD with faster convergence and less communication: Demystifying why model averaging works for deep learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 5693–5700.

Zhou, B.; Cui, Q.; Wei, X.-S.; and Chen, Z.-M. 2020. Bbn: Bilateral-branch network with cumulative learning for long-tailed visual recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 9719–9728.