Resource-Efficient Personalization in Federated Learning with Closed-Form Classifiers

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 6 March 2025, accepted 26 March 2025, date of publication 31 March 2025, date of current version 14 April 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3556587

RESEARCH ARTICLE

Resource-Efficient Personalization in Federated Learning With Closed-Form Classifiers

EROS FANÌ[1], RAFFAELLO CAMORIANO[1,2], BARBARA CAPUTO[1,3], AND MARCO CICCONE[4]

[1]Department of Control and Computer Engineering, Polytechnic University of Turin, 10129 Turin, Italy
[2]Istituto Italiano di Tecnologia, 16163 Genoa, Italy
[3]CINI Consortium, 00185 Rome, Italy
[4]Vector Institute, Toronto, ON M5G 0C6, Canada

Corresponding author: Eros Fanì (eros.fani@polito.it)

This study was carried out within the FAIR - Future Artificial Intelligence Research and received funding from the European Union Next-GenerationEU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) – MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.3 – D.D. 1555 11/10/2022, PE00000013). This manuscript reflects only the authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

ABSTRACT Statistical heterogeneity in Federated Learning (FL) often leads to client drift and biased local solutions. Prior work in the literature shows that client drift particularly affects the parameters of the classification layer, hindering both convergence and accuracy. While Personalized FL (PFL) addresses this by allowing client-specific models, it can overlook valuable global knowledge. This paper introduces Federated Recursive Ridge Regression (`Fed3R`), a fast and efficient method to construct a closed-form classifier that effectively incorporates global knowledge while being inherently robust to statistical heterogeneity. `Fed3R` leverages a pre-trained feature extractor and a recursive ridge regression formulation to achieve exact aggregation of local classifiers and recover the centralized solution. We demonstrate that `Fed3R` serves as a robust initialization for further fine-tuning with various FL and PFL algorithms, accelerating convergence and boosting performance. Furthermore, we propose Only Local Labels (`OLL`), a novel PFL technique that simplifies local classifiers by focusing only on locally relevant classes, preventing misclassifications and improving efficiency. Our empirical evaluation on real-world cross-device datasets shows that `Fed3R`, combined with `OLL`, significantly improves performance and reduces training costs in heterogeneous FL and PFL scenarios.

INDEX TERMS Federated learning, personalized federated learning, pre-trained models, ridge regression, closed-form classifiers.

I. INTRODUCTION

Federated Learning (FL) [1] has emerged as a promising paradigm for training machine learning models on decentralized data sources, such as smartphones and IoT devices, while preserving user privacy. By leveraging local data from these devices, FL enables the development of expressive models without requiring the transmission of sensitive information.

However, training models in federated settings presents unique challenges, particularly in cross-device scenarios where thousands or even millions of devices participate with limited availability, resources, and heterogeneous data distributions [2]. This statistical heterogeneity, arising from variations in user habits [3], geographical locations [4], [5], and device preferences, can lead to client drift [6]. This issue occurs when local updates become biased towards specific clients' data, hindering the effectiveness of aggregation strategies like `FedAvg` [1] and negatively impacting convergence speed, communication efficiency, and model performance.

To address these challenges, various FL algorithms have been proposed, including methods that employ regularization [7], [8], [9] or aim to align local updates with the direction in the global loss landscape [6], [10], [11]. Another approach is Personalized Federated Learning (PFL) [5], a different optimization problem that focuses on training

The associate editor coordinating the review of this manuscript and approving it for publication was Jeon Gwanggil.

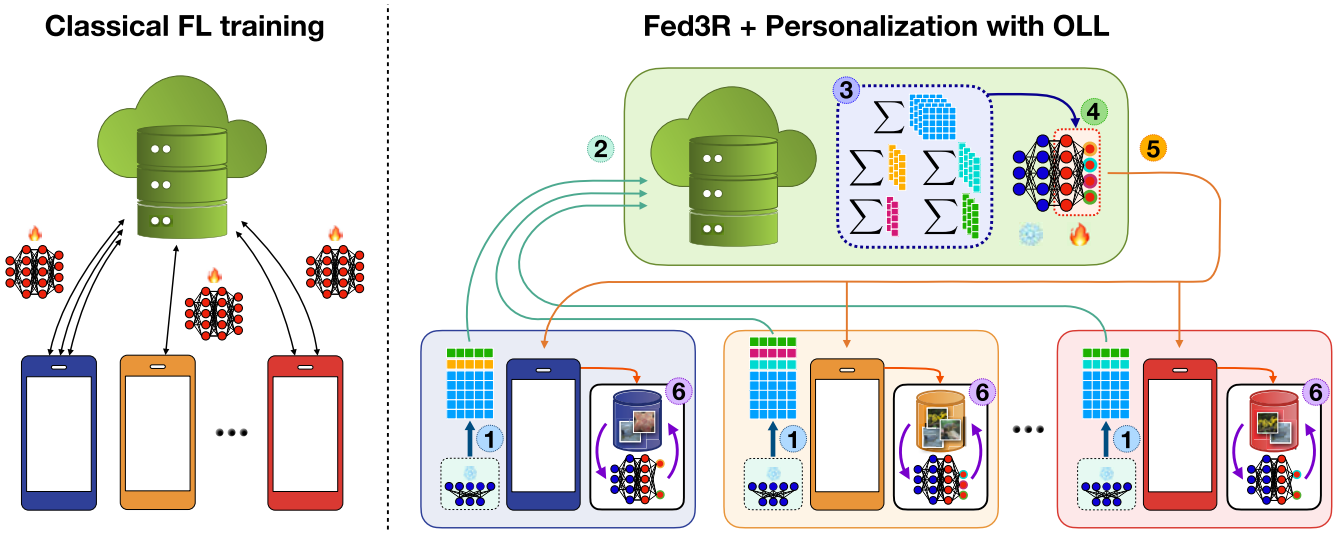

FIGURE 1. On the left, we show the classical FL training pipeline. Here, multiple arrows between clients and the server indicate that clients can be potentially sampled multiple times during training, resulting in numerous exchanges of information and high communication costs; each arrow represents one exchange of information between the clients and the server, potentially in different rounds. On the right, we present an overview of our `Fed3R` and `OLL` methods. The model is represented as a neural network. In blue and red, we indicate the fixed and learnable layers of the model. Classical FL training requires multiple rounds of communication. Conversely, `Fed3R` requires only one upstream communication from the clients to the server. Steps 1-4 refer to the `Fed3R` algorithm, while steps 5-6 refer to the `OLL` method. 1) Each client extracts the local `Fed3R` statistics using the pre-trained feature extractor. 2) Each client shares its `Fed3R` statistics with the server. 3) The server aggregates the received statistics. 4) The server uses the aggregated statistics to compute the optimal parameters for the `Fed3R` classifier. 5) The server shares the `Fed3R` classifier with the clients. 6) The clients filter the parameters of the classifier, keeping only the ones associated with the local classes, and fine-tune the classifier locally.

personalized local models tailored to individual client's data distributions [3], [12], [13], [14]. Nevertheless, because of statistical heterogeneity, PFL algorithms face challenges similar to FL ones in effectively extracting global knowledge from clients participating in the federation, which is needed for effective personalization. As a result, clients tend to prioritize their own local data, often missing out on valuable global knowledge that could be obtained from other clients.

Recent work has highlighted the benefits of leveraging pre-trained feature extractors in FL to mitigate the negative effects of heterogeneity and accelerate convergence [15], [16]. However, the classifier, often trained from scratch, remains susceptible to instability in cross-device scenarios due to partial client participation and non-independent and identically distributed (non-i.i.d.) data. This can lead to a detrimental cycle where a weak classifier deteriorates the feature extractor's updates, which in turn provides worse feature maps to the classifier, hindering the training process [17], [18]. This issue is further exacerbated in the final prediction layer by recency bias (or catastrophic forgetting) [19], [20], [21], [22], where the model becomes overly influenced by the most recently active clients, particularly in cross-device FL with non-i.i.d. and class-imbalanced data [23], [24], [25], [26].

To address these challenges and maximize the benefits of pre-trained feature extractors in both FL and PFL, we propose Federated Recursive Ridge Regression (`Fed3R`), a novel method that constructs a classifier robust to statistical heterogeneity. `Fed3R` leverages an online formulation of Ridge Regression (RR) to efficiently compute the classifier in a closed form, exploiting feature maps generated by a pre-trained feature extractor. Each client computes and sends its local RR statistics to the server, which aggregates them to obtain a classifier equivalent to the centralized RR solution. This approach ensures exact aggregation, requires only a single communication round per client, and provides immunity to client sampling ordering and statistical heterogeneity.

Furthermore, we demonstrate that initializing the classifier with `Fed3R` weights and subsequently fine-tuning the model with any FL algorithm can significantly accelerate training and improve performance in both FL and PFL. Our experiments on real-world cross-device datasets [27], [28], [29] show that `Fed3R` initialization leads to substantial reductions in training time, communication, and computations while achieving competitive or even superior performance compared to expensive initialization methods, especially in highly heterogeneous scenarios. By providing a robust and stable initialization, `Fed3R` helps break the vicious cycle between the classifier and feature extractor, mitigating the negative effects of recency bias and destructive interference during the aggregation phase.

Finally, to further enhance the effectiveness of personalization in PFL, we introduce Only Local Labels (`OLL`), a novel strategy that simplifies the local classifier by filtering out classes not present in the client's local dataset. This approach prevents potential classification errors due to irrelevant classes and improves the efficiency of local fine-tuning.

Our empirical results demonstrate consistent and substantial improvements achieved by OLL, particularly when applied on a classifier initialized with the Fed3R optimal parameters and fine-tuned for some rounds of FL training. A complete overview of our methods is shown in Figure 1.

A. CONTRIBUTIONS

- We propose Fed3R, a simple and efficient algorithm that reduces communication and computational costs by learning a linear classifier for heterogeneous cross-device FL and PFL that is immune to statistical heterogeneity. Therefore, it is not affected by data recency bias [18], [23] and destructive interference [30].
- We show that Fed3R parameters can be further fine-tuned using any FL or PFL strategy. Additionally, we show that the Fed3R initialization greatly reduces the time to convergence and improves the quality of the feature maps.
- We propose OLL, a technique that simplifies the local clients' model personalization problem, improving final accuracy by a large margin. Our experiments demonstrate that the best results are obtained by using both our Fed3R and OLL methods together.
- We empirically demonstrate the efficacy of our methods on the Google Landmarks [27] and iNaturalist [28] large-scale, heterogeneous, cross-device datasets, which counts thousands of clients and classes with skewed data distributions.

B. EXTENSION DETAILS

The present paper is an extension to our previous manuscript, "Accelerating Heterogeneous Federated Learning with Closed-form Classifiers" [31]. We summarize the elements of this work in Figure 2, highlighting the new components and methods with a colored background. In particular, compared to the previous version:

① We modify the Fed3R algorithm, reducing communication and computational costs compared to the previous version. Further details on this will be given in Section IV.

② We focus, in addition to FL, on the PFL scenario. In particular, we show that the Fed3R classifier can be easily repurposed to PFL, providing a robust initialization. Additionally, we propose a new framework for a thorough empirical analysis of different methods in the PFL setting, conceptually dividing the possible strategies into four, optional, sequential phases. During the first, the classifier can be initialized with a closed-form classifier, such as the Fed3R classifier. In the second phase, classical FL training is performed. Finally, in the third and fourth phases, we run partial and full personalization algorithms (the distinction between partial and full personalization is presented in Section III-C). The objective is to understand the contribution of each single phase to the overall training procedure. Thanks to our framework, we are able to understand the best, optimal approach that achieves the highest possible performance, minimizes communication and computational costs, and increases convergence speed. Further details are exposed in Section V-A.

③ We introduce the OLL method for full personalization. We show that combining Fed3R classifier initialization with OLL makes it possible to achieve remarkable results, compared with the baselines, in highly statistically heterogeneous PFL scenarios while reducing costs.

④ We introduce a new heterogeneous, cross-device, PFL split (*i.e.*, a new partition of a dataset into local clients' datasets) for the iNaturalist dataset [28]: Pers-Geo-100. We construct this split following a similar protocol as the one followed by [32] for the Pers-Users-160K for the Google Landmarks dataset [27]. Further details are presented in Section V-B.

II. RELATED WORKS

A. STATISTICAL HETEROGENEITY AND RECENCY BIAS IN FEDERATED LEARNING

While FL methods are effective when dealing with i.i.d. data, handling statistical heterogeneity and real-world scenarios remains a significant challenge [2]. Realistic private data can be biased by factors such as personal habits and geographical locations [2], [5], [29], leading to variations among clients in terms of observed classes, domains, and dataset sizes. This bias causes the *client drift* phenomenon [6], where local models converge towards different minima, deviating from the global direction and resulting in noisy, unstable learning updates [25], [33].

Several strategies have been adopted in the literature to address the issue of heterogeneity. One is to limit the divergence of local models by applying regularization techniques [7], [8] or stochastic variance reduction methods [6], [34] to align local model updates. Other approaches leverage the history of previous updates by incorporating momentum or adaptive optimizers during the server aggregation [35], [36], [37], or introduce client-side momentum [11], [38], [39] to reduce client drift and guide local updates in the direction of the global model. Other works aim to replicate the behavior of models trained on i.i.d. data by combining stochastic variance reduction and client-side momentum [38].

However, statistical heterogeneity and client drift affect the different layers of the models with various severity levels. In particular, [30] shows that clients' bias towards local data distributions is more pronounced in the deeper layers of the model, particularly in the last one, the prediction head. Other studies have noted that biased classifiers and misaligned features create a harmful cycle [17], [18]. For instance, [40] suggests that learning the feature extractor and the classifier in two separate phases may be beneficial in FL, as observed in other fields [41], [42]. Moreover, [16] points out that discriminative features are essential for effectively training models, as convergence speed improves when gradients are better aligned. Similarly to these works, [43], [44], [45], [46],

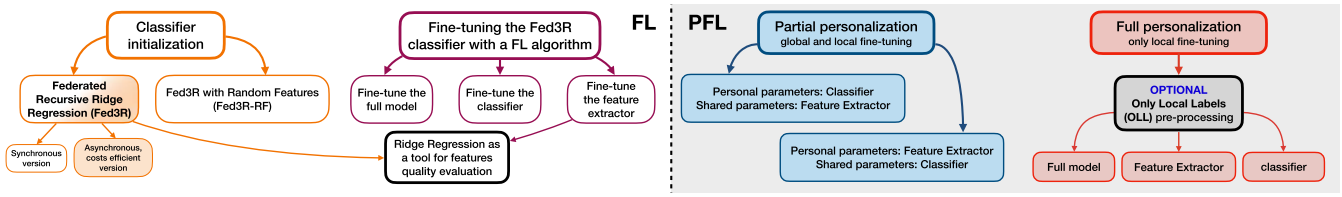

FIGURE 2. Summary of the main contributions of this work and its main components. On the left, we present the main contributions related to FL and, on the right, those related to PFL. Extensions with respect to [31] are highlighted with boxes with a colored background.

[47] highlight that, when clients train the model on their local data, the final layer diverges the most, becoming highly skewed toward the local distribution.

Several works [16], [30], [48] propose to mitigate the data recency bias by retraining only the classifier on the server exploiting virtual features. However, this approach remains sub-optimal as it relies on the quality of the feature representation and the generative process, which could negatively impact the retrained classifiers. Conversely, the authors of [49] propose to address this problem with a two-stage algorithm that uses a fixed, random classifier and trains only the feature extractor. Then, each client fine-tunes its own classifier using local data, developing personalized solutions. More recently, [18] proposes a fixed synthetic classifier motivated by the simplex geometry of the logits space induced by neural collapse [50], while [51] introduces an incremental aggregation scheme for recurrent classifiers on time-series data.

Finally, [52] proposes `FedNCM`, an algorithm that employs a Nearest Class Mean (NCM) classifier to avoid gradient updates. Similarly to our `Fed3R` classifier, the `FedNCM` classifier is not gradient-based. While `FedNCM` constructs a global classifier by using class prototypes computed from local client data, our `Fed3R` algorithm leverages local clients statistics to construct a ridge regression classifier equivalent to the centralized solution. Although `FedNCM` may be effective on simpler datasets, in Section V-C and Appendix D, we demonstrate its weaknesses in realistic scenarios, where it is always outscored by our `Fed3R` classifier.

B. RIDGE REGRESSION IN DISTRIBUTED AND FEDERATED LEARNING

We now review prior works concerning RR in federated and distributed settings. In [53], an RR model is trained in a federated scenario with only two clients via knowledge distillation. RR has also been applied to the *Vertical FL* (V-FL) setting, where the feature space is partitioned across clients that share the same samples [54], [55]. On the other hand, our work targets *Horizontal FL* (H-FL) [56], in which clients share the same feature space while sample instances can differ. The distinct partitioning criteria between V-FL and H-FL result in fundamentally different federated and distributed RR approaches. In particular, client drift and statistical heterogeneity are major issues in H-FL, while this is not the case for V-FL. RR variants have been proposed in the distributed learning and optimization fields [57], [58], [59]. Still, these settings are significantly different from FL, since they lack privacy requirements and usually assume i.i.d.-distributed data. In this manuscript, we focus on RR in H-FL cross-device settings. We leverage its exact incremental formulation to overcome the issue of statistical heterogeneity in the aggregation step while improving computational and communication efficiency.

C. PRE-TRAINED MODELS IN FEDERATED LEARNING

Recent work has explored the use of pre-trained representations in FL, demonstrating their potential to reduce training time, mitigate the negative effects of data heterogeneity, and enhance performance [15], [16]. However, effectively and efficiently fine-tuning these representations and training the classifier remains a challenge, particularly in highly heterogeneous and cross-device settings. In such settings, the classifier is highly susceptible to data recency bias [30], where data imbalance and partial client participation can skew the model towards the data from the most recently encountered clients. This bias can trigger a detrimental feedback loop, negatively impacting the quality of the learned representation [18].

Furthermore, the advent of Foundation Models has fueled interest in adapting these powerful models to cross-device FL settings [60], [61], [62], [63]. This adaptation often involves parameter-efficient fine-tuning techniques based on Low-Rank Approximation (LoRA) [64], [65], [66] to address the challenges of limited resources and communication constraints inherent in federated environments.

In this work, we focus on efficiently leveraging pre-trained representations to accelerate training and facilitate knowledge reuse in both federated and personalized learning scenarios. Specifically, we introduce `Fed3R`, a novel algorithm that utilizes a pre-trained feature extractor and constructs a classifier using exact aggregation, eliminating the need for gradient-based learning. `Fed3R` addresses the challenges of classifier instability and data recency bias, enabling robust and efficient adaptation of pre-trained models to heterogeneous and cross-device settings.

D. PERSONALIZED FEDERATED LEARNING

Given the challenges of learning global models in heterogeneous settings, some works [2], [3], [5] point out that tailoring models to the specific needs of individual clients

while benefiting from shared knowledge across the network could be a possible alternative solution, introducing the PFL setting. Several approaches to PFL have been proposed, such as model personalization techniques for multi-task learning [13], [67] and meta-learning [5], [68], clustered FL methods [14], [69], [70], [71], [72], or regularization strategies [73], [74], [75], [76]. In particular, `Ditto` [74] adds a regularization term similar to the one of [8] to approach local model updates to the global model, while `pFedMe` [73] uses Moreau envelopes to decouple local and global model updates, allowing for client-specific customization without losing the benefits of collaborative learning.

Other approaches tackle the PFL problem by dividing the model parameters into two groups. Using the nomenclature of [32], the *shared parameters* are aggregated globally in a traditional FL manner, while the *personal parameters* are client-specific and remain local to the clients, capturing their individual distribution. In particular, [12], [77], [78] propose to learn the two groups together, while [79], [80] learn the two groups separately updating the personal parameters before the shared ones. The authors of [32] refer to the two strategies as `FedSim` and `FedAlt` respectively. In this work, we adopt the same shared vs. personal parameters structure.

III. BACKGROUND

This section provides the necessary background and notations preparatory for Section V. In particular, Section III-A provide a concise overview of both FL and PFL and the most common strategies to address such scenarios, while Section III-D introduces the fundamental concepts of Ridge Regression (RR). A summary of the notation used in this manuscript is proposed in Table 1.

A. FEDERATED LEARNING SETTING AND NOTATION

Consider a standard FL scenario with a set of clients $\mathcal{K}$, where $|\mathcal{K}| = K$, and a central server $\mathcal{S}$ that coordinates the training process. In cross-device FL, K is typically assumed to be large, potentially reaching millions or even billions of devices [2]. Each client $k \in \mathcal{K}$ possesses a local dataset $\mathcal{D}_k$ containing $n_k = |\mathcal{D}_k|$ samples, where n_k could be small and significantly vary across clients. Crucially, neither the server $\mathcal{S}$ nor any other client $j \neq k$ can access the local dataset $\mathcal{D}_k$.

Each local dataset $\mathcal{D}_k$ consists of n_k samples $(x, y) : x \in \mathcal{X}, y \in \mathcal{Y}$, where $\mathcal{X}$ is the input space and $\mathcal{Y}$ is the output space. The overall dataset split across clients can be denoted by $\mathcal{D} = \bigcup_{k\in\mathcal{K}} \mathcal{D}_k$, with a total of $|\mathcal{D}| = \sum_{k\in\mathcal{K}} n_k = n$ samples. Data among clients is assumed to be non i.i.d., reflecting the heterogeneity often encountered in real-world FL scenarios. Additionally, all the clients $k \in \mathcal{K}$ and the server $\mathcal{S}$ have access to a *model* $\mathcal{M} : \mathcal{X} \to \mathcal{Y}$ parameterized by parameters $\theta \in \Theta$, where Θ is the parameters space.

In this work, we consider models $\mathcal{M}$ that can be decomposed into a *classifier* $\psi : \mathcal{Z} \to \mathcal{Y}$ and a *feature extractor* $\varphi : \mathcal{X} \to \mathcal{Z}$, such that $\mathcal{M} = \psi \circ \varphi$. The feature extractor φ maps the input space $\mathcal{X}$ to a latent feature space $\mathcal{Z} \subseteq \mathbb{R}^d$, where $d \in \mathbb{N}$, and the classifier ψ maps the latent space to the output space $\mathcal{Y} \subseteq \mathbb{R}^C$. The classifier ψ is parameterized by $\theta_\psi \in \mathbb{R}^{d\times C}$, while the feature extractor φ is parameterized by the remaining model parameters, denoted as θ_φ. For brevity, we will refer to the *latent feature space* $\mathcal{Z}$ simply as *latent space*.

TABLE 1. List and description of the main symbols introduced in this manuscript.

Notation	Description
$\mathcal{S}$	server which orchestrates the training procedure
$\mathcal{K}$	set of clients
k	specific client $\in \mathcal{K}$
$K \in \mathbb{N}$	total number of clients
$\mathcal{D}_k$	dataset of client k
$n_k \in \mathbb{N}$	number of samples in $\mathcal{D}_k$
$\mathcal{C}$	set of all the possible classes
$\mathcal{C}_k$	set of all the classes of client k
$p \in \mathbb{N}$	input space dimensionality
$C \in \mathbb{N}$	total number of distinct classes
$C_k \in \mathbb{N}$	number of distinct classes of client k
$d \in \mathbb{N}$	latent space dimensionality
$\varphi : \mathcal{X} \subseteq \mathbb{R}^p \to \mathcal{Z} \subseteq \mathbb{R}^d$	feature extractor
$\psi : \mathcal{Z} \subseteq \mathbb{R}^d \to \mathcal{Y} \subseteq \mathbb{R}^C$	classifier
$\mathcal{M} = \psi \circ \varphi$	entire model
θ	parameters of the entire model
θ_φ	parameters of the feature extractor
$\theta_\psi \in \mathbb{R}^{d\times C}$	parameters of the classifier
$X_k \in \mathbb{R}^{n_k\times p}$	stacked input vectors of client k
$Z_k = \varphi(X_k) \in \mathbb{R}^{n_k\times d}$	stacked latent feature vectors of client k
$Y_k = \psi(Z_k) \in \mathbb{R}^{n_k\times C}$	stacked one-hot encoding vectors associated to ground-truth labels of client k
$A_k \in \mathbb{R}^{d\times d}$	variance-covariance matrix of the latent feature vectors Z_k of client k
$b_k \in \mathbb{R}^{d\times C}$	$Z_k^\top Y_k$
$b_k^c \in \mathbb{R}^d$	column of the matrix b_k associated to class c
$t \leq K \in \mathbb{N}$	number of clients that have already sent their statistics to the server
$A_t \in \mathbb{R}^{d\times d}$	global variance-covariance matrix with the statistics from t clients
$b_t \in \mathbb{R}^{d\times C}$	global version of the b_k matrix updated with the statistics from t clients
$W_t \in \mathbb{R}^{d\times C}$	provisional `Fed3R` classifier parameters computed with the statistics from t clients
$W^* = W_K$	Optimal `Fed3R` classifier parameters computed with the statistics from all the K clients
$W^c \in \mathbb{R}^d$	column of W associated to class c
$\hat{\theta}_\psi \in \mathbb{R}^{d\times C_k}$	parameters of the classifier after `OLL` pre-processing

B. FL AND PFL OBJECTIVES

In FL, the primary goal is to train a single global model $\mathcal{M}$ that performs well across all clients. This involves finding the optimal global parameters $\theta^* \in \Theta$ that minimize the empirical objective function:

$$F(\theta) = \sum_{k\in\mathcal{K}} \frac{n_k}{n} \mathcal{L}_k(\theta), \tag{1}$$

where $\mathcal{L}_k(\theta) = \sum_{(x,y)\in\mathcal{D}_k} \ell(\mathcal{M}(x;\theta), y)$ represents the local empirical risk for client k. This risk is calculated using a suitable loss function ℓ, such as the cross-entropy loss.

In contrast, PFL focuses on finding distinct, optimized local models for each client. The objective in PFL is to determine the optimal local parameters $\theta_k^* \in \Theta$ for each client $k \in \mathcal{K}$ that minimize the following empirical objective function:

$$F(\{\theta_k\}_{k\in\mathcal{K}}) = \sum_{k\in\mathcal{K}} \frac{n_k}{n} \mathcal{L}_k(\theta_k). \tag{2}$$

This approach allows for greater flexibility and to adapt to the unique characteristics of each client's data distribution.

C. TRAINING STRATEGIES

FL relies on an iterative training procedure consisting of $T \in \mathbb{N}$ optimization rounds. In each round $t \in [T]$, the server $\mathcal{S}$ selects a random subset of clients $\mathcal{K}' \subseteq \mathcal{K}$ and sends them the current global parameters θ^t. Each client $k \in \mathcal{K}'$ initializes its local parameters with $\theta_k^t = \theta^t$ and performs local training on its dataset $\mathcal{D}_k$ for a certain number of iteration. This produces the updated parameters θ_k^{t+1}, which are then transmitted back to the server. Finally, $\mathcal{S}$ aggregates the received updates to compute the new global parameters θ^{t+1} for the next round.

For PFL, following the framework in [32], we consider two distinct strategies: *partial personalization* and *full personalization*. In **Partial Personalization (`PP`)**, the local parameters θ_k of each client $k \in \mathcal{K}$ are divided into two sets: *shared parameters* $u_k \in \mathcal{U}$ and *personal parameters* $v_k \in \mathcal{V}$, where $u_k \cup v_k = \theta_k$ and $u_k \neq \emptyset$, $v_k \neq \emptyset$. At the beginning of each round $t \in [T]$, the server sends the current global shared parameters u^t to the sampled clients $k \in \mathcal{K}'$, which initialize their local shared parameters with $u_k^t = u^t$. The clients then locally optimize both their shared and personal parameters, resulting in updated parameters $\theta_k^{t+1} = u_k^{t+1} \cup v_k^{t+1}$. However, only the updated shared parameters u_k^{t+1} are communicated back to the server, while the personal parameters v_k^{t+1} are retained locally. The server then aggregates the received u_k^{t+1} to obtain the updated global shared parameters u^{t+1} for the next round. In contrast, **Full Personalization (`FP`)** represents the scenario where $u_k = \emptyset$ and $v_k \subseteq \theta_k$ for all $k \in \mathcal{K}$. If $\theta_k \setminus v_k \neq \emptyset$, the parameters $\theta_k \setminus v_k$ are not updated during training, *i.e.*, they are fixed. The case where $v_k = \theta_k$ can be viewed as a special case of partial personalization where all parameters are personal. In this strategy, no communication is required between the clients and the server, as each client trains its model independently on its local data. Consequently, no rounds are necessary in this setting.

D. RIDGE REGRESSION

Consider a *centralized*, supervised learning setting with a dataset $\mathcal{D}$ composed of n samples (x, y), where $x \in \mathbb{R}^p$ are the inputs and $y \in \mathbb{R}^C$ are the targets, $p, C \in \mathbb{N}$. All the samples can be accessed simultaneously. One-vs-rest linear predictors such as *least-squares regressors* [81], [82], [83] admit a closed-form solution which can be computed efficiently. However, these predictors are generally prone to overfitting [84]. To mitigate this issue, an $\mathcal{L}_2$ regularization term (controlled by a Tikhonov hyper-parameter $\lambda \in \mathbb{R}^+$) can be incorporated into the objective function, leading to the RR [85] problem:

$$W^* = \underset{W \in \mathbb{R}^{p \times C}}{\arg\min} \|Y - XW\|^2 + \lambda \|W\|^2, \tag{3}$$

where $X \in \mathbb{R}^{n \times p}$ and $Y \in \mathbb{R}^{n \times C}$ are the matrices of the n stacked inputs and targets. The solution $W^* \in \mathbb{R}^{p \times C}$ provides the optimal parameters for the linear predictor $\psi(x; W) = W^\top x$ which can be computed in closed-form as:

$$W^* = (X^\top X + \lambda I_p)^{-1} X^\top Y, \tag{4}$$

where I_p is the $p \times p$ identity matrix.

Since $X^\top X + \lambda I_p \succ 0$ for any $\lambda > 0$, no additional assumptions on the rank or the dimensions of the matrix X are required for the optimal solution W^* to exist [85]. Furthermore, the RR solution converges in probability to the optimal Bayes classifier as n tends to infinity [86], [87], [88]. Finally, RR can be applied to classification problems, as demonstrated in [83]. By representing the C categories with stacked one-hot encoded vectors in the target matrix Y, the classification problem can be effectively treated as a regression problem.

Recursive Ridge Regression. In many practical scenarios such as Continual and Incremental Learning [41], [89], data becomes available sequentially rather than all at once. Similarly, in decentralized settings such as FL, data are not available in the same location but are distributed across clients. For this reason, Eq. (4) cannot be directly applied to obtain the optimal linear parameters W^*. However, Recursive RR [90] offers a practical way to update the optimal solution as new data are collected or shared. This method leverages the linearity of Eq. (4) and employs techniques like Sherman-Morrison-Woodbury or Cholesky updates [91], [92] for efficient computation. Specifically, if the samples are presented sequentially in batches $\{(X_t, Y_t)\}_{t=1}^T$, $T \in \mathbb{N}$, the centralized solution of Eq. (4) can be obtained recursively as follows:

$$\begin{cases} M_{t+1} &= P_t X_{t+1}^\top (\lambda I_{n_{t+1}} + X_{t+1} P_t X_{t+1}^\top)^{-1} \\ P_{t+1} &= P_t - M_{t+1} X_{t+1} P_t \\ W_{t+1} &= M_{t+1}(Y_{t+1} - X_{t+1} W_t) \end{cases} \tag{5}$$

where n_t is the number of samples in batch t, $P_0 = \frac{1}{\lambda} I_p$ and $W_0 = 0_{p \times C}$, where $0_{p \times C}$ is a $p \times C$ matrix of zeros.

While Eq. (5) provides an efficient solution for incremental learning, it can be noticed that this is not suitable for privacy-constrained applications where data are not available in a central location and cannot be communicated. This is because Eq. (5) requires direct access to the data batches (X_{t+1}, Y_{t+1}) to compute the updated M_{t+1}, P_{t+1}, and W_{t+1} matrices. Such direct access violates the core principle of decentralized, privacy-constrained environments, where data remains local.

To address this challenge, we introduce a novel method in Section IV-A that leverages the linear properties of RR to enable learning in decentralized settings. Our proposed approach constructs an optimal, equivalent classifier without requiring the transmission of private data, but only aggregated local matrices.

IV. METHOD

This section provides a formal description of our proposed methods. Section IV-A introduces `Fed3R`, a novel algorithm that enables solving the RR problem for classification in

a federated setting, guaranteeing a solution mathematically equivalent to the centralized one. Section IV-B proposes an effective way of using Fed3R as a robust initialization for further fine-tuning in both FL and PFL scenarios. Finally, Section IV-C introduces OLL, a novel strategy for PFL that simplifies the local classifier by filtering out classes not present in the local data distribution. Notably, OLL can be applied to any classifier, including Fed3R, which can be further fine-tuned using only local class labels for improved final performance.

A. FEDERATED RECURSIVE RIDGE REGRESSION (FED3R)

In this section, we present Fed3R, showing how RR can be efficiently repurposed for the FL setting with a recursive formulation.

Our objective is to determine the optimal parameters θ_ψ^* for the classifier ψ in a federated setting where clients collaboratively contribute to the solution. We leverage a pre-trained feature extractor φ to map the input data into the latent space $\mathcal{Z} \subseteq \mathbb{R}^d$. By exploiting the closed-form solution of RR and considering the latent space $\mathcal{Z}$ as the input space, we can reformulate the centralized RR solution in Eq. (4) as:

$$W^* = (A + \lambda I_d)^{-1} b, \tag{6}$$

where $A = Z^\top Z = \sum_{(x,y)\in\mathcal{D}} \varphi(x)\varphi(x)^\top \in \mathbb{R}^{d\times d}$ is the covariance matrix of samples in the latent space, $Z \in \mathbb{R}^{n\times d}$ is the matrix of mapped input samples with rows $Z_i = \varphi(X_i)$ and $b := Z^\top Y = \sum_{(x,y)\in\mathcal{D}} \varphi(x) e_y^\top \in \mathbb{R}^{d\times C}$ is the sum of the samples aggregated per class, as e_y represents the one-hot encoding vector for the class y.

The complete Fed3R algorithm is shown in Algorithm 1 and pictorically described in Figure 3. The core idea of Fed3R is to treat each local dataset $\mathcal{D}_k$ as separate batches and recursively update the RR solution. However, a direct application of the recursive RR formulation in Eq. (5) would require clients to share their raw data with the server, violating privacy constraints.

To address this challenge, Fed3R leverages the structure of the A and b matrices that can be cumulatively updated without re-computing them from scratch [82], [93]. Indeed, these matrices can be expressed as the sum of individual contributions from each client's local dataset $\mathcal{D}_k$:

$$A = \sum_{(x,y)\in\mathcal{D}} \varphi(x)\varphi(x)^\top = \sum_{k\in\mathcal{K}} \sum_{(x,y)\in\mathcal{D}_k} \varphi(x)\varphi(x)^\top = \sum_{k\in\mathcal{K}} A_k, \tag{7}$$

$$b = \sum_{(x,y)\in\mathcal{D}} \varphi(x) e_y^\top = \sum_{k\in\mathcal{K}} \sum_{(x,y)\in\mathcal{D}_k} \varphi(x) e_y^\top = \sum_{k\in\mathcal{K}} b_k. \tag{8}$$

Notably, Eqs. (7) and (8) are true for all the possible partitions of $\mathcal{D}$ into subsets $\mathcal{D}_k$ such that $\mathcal{D} = \bigcup_{k\in\mathcal{K}} \mathcal{D}_k$ is the union of the local datasets $\mathcal{D}_k$, with $\mathcal{K}$ the set of all the clients. For this reason, once all clients communicate their A_k and b_k statistics to $\mathcal{S}$, the server can compute the Fed3R solution equivalent to the centralized RR solution irrespective of the federated clients' split, making the Fed3R algorithm immune to statistical heterogeneity and invariant to the order the clients share their statistics with the server.

Furthermore, to minimize communication overhead, clients can optimize the transmission of b_k. Unlike the approach in [31], where clients transmitted the full b_k matrix, here we propose a more efficient strategy. Instead of sending the full matrix, which may contain many zero columns associated with classes not present in the local dataset, clients only transmit the non-zero column vectors b_k^c for $c \in \mathcal{C}_k$, where $\mathcal{C}_k \subseteq \mathcal{C}$ is the set of classes present in client k's local dataset. Once the server receives the column vectors b_k^c, it sums them in the corresponding column c of the global b matrix. As empirically demonstrated in Section IV-A, this new strategy leads to substantial communication savings by eliminating the need to compute the full b_k matrix on clients and share it with the server.

Once the server $\mathcal{S}$ has received the statistics A_k and b_k^c from all the clients and updates the classifier parameters, the Fed3R solution is mathematically equivalent to the optimal centralized classifier W^*. Consequently, the Fed3R classifier inherits all the properties and guarantees of the centralized RR classifier.

Finally, to address the class imbalance of the global distribution in realistic scenarios, we normalize the final classifier parameters W by dividing each column by its norm: $W^c \leftarrow W^c / \|W^c\|, \forall c \in \mathcal{C}$. This normalization acts as a reweighing mechanism, as classes with more data points tend to have larger magnitudes in W, and this scaling ensures that under-represented classes are considered for the prediction.

Unlike gradient-based FL algorithms, Fed3R does not rely on assumptions such as bounded variance of stochastic gradients or smoothness of clients' loss landscapes [2], [6], [7], [38] making it readily applicable to real-world scenarios. Moreover, conversely to most FL algorithms that require multiple samplings of the same clients, Fed3R only needs each client to share its statistics with the server only once, saving communication costs and accelerating training. Namely, each client needs to be sampled only once.

In Appendix A, we introduce Fed3R with Random Features (Fed3R-RF), a variant of Fed3R that approximates the Kernel RR solution using Random Features [94] to handle non-linearities in the input space.

Fed3R theoretical properties

Below, we collect the six theoretical properties of Fed3R:

① **The Fed3R solution is mathematically equivalent to the optimal centralized RR solution.**
This is a consequence of Eqs. (7) and (8), which demonstrates that the matrices A and b, necessary to compute the final classifier parameters W^*, can be recovered by the local statistics of the clients exactly.

② **The Fed3R algorithm is immune to statistical heterogeneity.**
This is a consequence of Property 1. Since Fed3R recovers the original centralized RR solution exactly, the particular split of the samples across the clients, which

may cause various levels of statistical heterogeneity, and the number of clients itself, is irrelevant.

③ **The Fed3R solution is invariant to the order in which the clients share their statistics with the server.**
This is a consequence of Property 1. Alternatively, it can be already observed by Eqs. (7) and (8).

④ **The Fed3R classifier inherits all the properties and guarantees of the centralized RR classifier.**
This is a consequence of Property 1. Since the Fed3R solution is equivalent to the centralized RR solution, Fed3R inherits all the RR properties, such as the properties mentioned in Section III-D.

⑤ **Fed3R does not rely on assumptions such as bounded variance of stochastic gradients or smoothness of clients' loss landscapes.**
These are common assumptions in other gradient-based FL algorithms [2], [6], [7], [38], limiting their theoretical insights' applicability in real-world scenarios. Fed3R does not suffer from this problem, as it does not necessitate these strong hypotheses.

⑥ **In Fed3R, each client needs to be sampled only once.**
Indeed, in contrast to most FL algorithms that require multiple samplings of the same clients, Fed3R only needs each client to share its statistics with the server only once, saving communication costs and accelerating training.

In Appendix B, we describe the privacy and security properties of Fed3R.

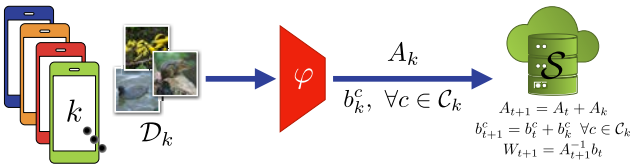

FIGURE 3. The Fed3R algorithm. A_k and b_k^c are computed according to Section IV-A.

B. FED3R FOR ROBUST CLASSIFIER INITIALIZATION

While Fed3R offers a simple and efficient solution to learn a classifier in a federated setting, effectively overcoming the issues of statistical heterogeneity, its performance is inherently tied to the quality of the pre-trained feature extractor. To further enhance the adaptability and performance of the model on the target task, we propose a federated fine-tuning stage with gradient-based FL algorithms leveraging Fed3R as robust initialization. In Section V, we demonstrate that Fed3R provides a powerful and stable initialization that can mitigate client drift and destructive interference during aggregation in heterogenous federated settings, which can arise from inconsistencies in local updates and lead to classifier divergence, breaking the detrimental cycle described in [17].

As the Fed3R classifier is already the optimal solution for the RR problem with the original feature extractor parameters, we fine-tune the model using the cross-entropy loss, which typically provides better performance for classification tasks [84]. To align the entropy of the predictions' distribution, we calibrate the model by tuning the temperature of the softmax function. For more details, we refer the readers to Section D.

Finally, **this extension work demonstrates that the Fed3R initialization can also benefit personalized learning scenarios** providing a robust starting point for both partial and full personalization algorithms in PFL, and enabling them to converge faster and achieve better performance.

Algorithm 1 - **Federated Recursive Ridge Regression** (**Fed3R**). In green and red are shown the differences with the ICML version [31] of this manuscript (green for the new version, red for the ICML version).

Def: Y_k is the matrix of stacked one-hot encoding vectors for all the classes of client k
Require:
Server $\mathcal{S}$, clients $\mathcal{K}$
Fixed pre-trained feature extractor $\varphi : \mathcal{X} \to \mathbb{R}^d$
$\mathcal{S}$ initializes $A_0 = \lambda I_d$ and $b_0 = 0_{d \times C}$
for each client $k \in \mathcal{K}$ in parallel **do**
  $Z_k = \varphi(X_k), \quad A_k = Z_k^\top Z_k$
  $b_k^c = \sum_{(x,y=c) \in \mathcal{D}_k} \varphi(x), \forall c \in \mathcal{C}_k \qquad (b_k = Z_k^\top Y_k)$
  Send A_k and $b_k^c\ \forall c \in \mathcal{C}_k$
end for
Every time $\mathcal{S}$ receives A_k and b_k:
  $A_{t+1} = A_t + A_k$
  $b_{t+1}^c = b_t^c + b_k^c\ \ \forall c \in \mathcal{C}_k \qquad (b_{t+1} = b_t + b_k)$
  $W_{t+1} = A_{t+1}^{-1} b_{t+1}$
  Normalize W: $W_{t+1}^c \leftarrow W_{t+1}^c / \left\| W_{t+1}^c \right\| \ \ \forall c \in \mathcal{C}$

1) FINE-TUNING STRATEGIES

In both FL and PFL scenarios, we explore three distinct fine-tuning strategies. The first possibility is to fine-tune both the feature extractor and the classifier. This strategy is suitable when both components can benefit from further adaptation to the target task. Another option involves fine-tuning only the classifier while keeping the pre-trained features fixed. This approach is appropriate when the pre-trained features are already robust and general enough for the target task. Finally, a third option consists of fine-tuning only the feature extractor while keeping the Fed3R classifier fixed. In Section V-C, we show that this strategy effectively mitigates destructive interference, especially in cross-device scenarios with high statistical heterogeneity, where the classifier is often the most susceptible to divergent updates [18], [30].

C. PERSONALIZATION WITH ONLY LOCAL LABELS (OLL)

While Fed3R can be used to efficiently integrate global knowledge from clients and obtain a classifier equivalent to the centralized RR, PFL focuses on optimizing models for individual clients' needs. In PFL, the goal is not to create a model suitable for all clients but to address each client's

Algorithm 2 - The Only Local Labels (OLL) algorithm.

Require:
Classifier parameters $\theta_\psi \in \mathbb{R}^{d\times C}$
Set of global classes $\mathcal{C}$, set of local classes $\mathcal{C}_k \subseteq \mathcal{C}$
Initialize $\hat{\theta}_\psi = 0_{d\times C_k}$
for each $c \in \mathcal{C}$ **do**
  if $c \in \mathcal{C}_k$ **then**
    $\hat{\theta}^c_\psi = \theta^c_\psi$
  end if
end for
Return $\hat{\theta}_\psi$

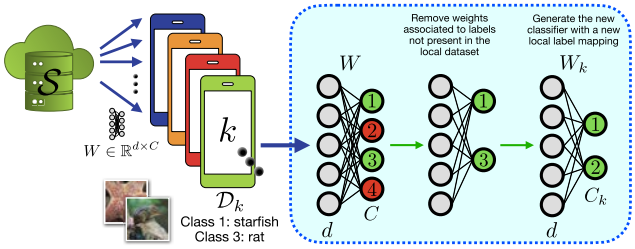

FIGURE 4. The OLL algorithm. In this example, the global mapping associates labels 1 and 3 to categories "duck" and "rat". In each client k, OLL filters out the columns of the classifier weights W associated with the classes not present in the local dataset $\mathcal{D}_k$ (2 and 4 in the example) and generates the new classifier parameterized by W_k with a local mapping. This classifier is already sufficient to improve the performance and can be further fine-tuned to the local distribution.

unique learning problem. Full personalization, where some or all model layers are fine-tuned locally, is a common approach to achieve this. However, directly fine-tuning the original classifier $\theta_\psi \in \mathbb{R}^{d\times C}$ can be inefficient, especially in highly heterogeneous settings where the local class set $\mathcal{C}_k$ is significantly smaller than the global class set $\mathcal{C}$.

In such cases, the classifier should ideally never need to predict classes outside the client's local distribution ($\mathcal{C} \setminus \mathcal{C}_k$), since we assume that clients operate with closed-set local datasets, as is common in many FL and PFL scenarios [1], [3], [6], [7], [38]. Yet, the original classifier, with its $d \times C$ shape, retains the potential to predict these irrelevant classes, hindering performance (e.g., class confusion) and slowing down training. This issue is particularly pronounced during the initial fine-tuning iterations, where the predicted probabilities may not be sufficiently biased towards the local classes.

To address this challenge, in this extension work, we introduce Only Local Labels (OLL), a novel PFL method designed to enhance personalization by simplifying the local classifier. As illustrated in Figure 4, OLL prunes the columns associated with the irrelevant classes ($\mathcal{C} \setminus \mathcal{C}_k$) from the classifier weights θ_ψ. This pruning generates a new, smaller classifier $\hat{\theta}_\psi \in \mathbb{R}^{d\times C_k}$ that focuses solely on the local classes $\mathcal{C}_k$ using a local label mapping. The complete OLL algorithm is shown in Algorithm 2.

This simplification offers several benefits. By preventing the prediction of irrelevant classes, OLL improves the efficiency and performance of the local classifier. Furthermore, it accelerates and simplifies subsequent local fine-tuning by simplifying the local problem. Moreover, reducing the classifier's dimensionality prevents interference from irrelevant classes. Importantly, OLL is a general technique that can be applied to any classifier, including the one obtained through Fed3R, and can be applied jointly to other PFL methods.

In Section V-E, we demonstrate that OLL pre-processing alone leads to performance improvements even without further fine-tuning. We also show that it further enhances the effectiveness and efficiency of subsequent local fine-tuning when combined with Fed3R. Finally, we validate the application of OLL by demonstrating that personalization inherently biases classifiers towards local classes, thereby justifying the use of OLL in scenarios with static client distributions. OLL simplifies the classification problem based on this insight, and is justified in scenarios where local classes are the primary focus and the distribution is assumed not to vary.

V. EXPERIMENTS

In this section, we outline the experimental setup and methodology used to evaluate Fed3R and OLL. These are compared against state-of-the-art baselines to assess their performance and validate the effectiveness of our proposed solutions.

Section V-C, presents our experiments within the FL setting. We first empirically demonstrate that Fed3R is immune to statistical heterogeneity, confirming our theoretical result, and we show that the new version of the algorithm is more cost-efficient compared to the version presented in our previous manuscript [31], as explained in Section IV-A. Furthermore, we show that Fed3R alone is sufficient to outperform baseline methods in highly heterogeneous scenarios, providing a strong initialization for the classifier's parameters.

Additionally, Section V-B introduces the datasets and the FL splits used in our experiments. Next, in Section V-A, we describe the structure of the experiments in this work, which consist of up to four distinct phases. Our findings indicate that not all phases are necessary, and the best strategy can depend on the available communication and computational budget.

Section V-D discusses the experiments conducted in the PFL scenario, analyzing both the partial and full personalization strategies. In both cases, Fed3R facilitates training, providing a robust initialization that mitigates the effects of client drift, reducing costs and enhancing final performance.

Finally, Section V-E demonstrates that OLL significantly improves accuracy without the need for further fine-tuning. Moreover, it simplifies local full personalization, guaranteeing higher final accuracies, as it prevents clients from predicting classes outside their local distribution.

In all our experiments, we employ the MobileNetV2 [95] architecture. The feature extractor θ_φ is pre-trained on ImageNet-1k [96]. In addition, Appendix section G describes in detail how the communication and computation costs of the algorithms are calculated.

A. EXPERIMENTS PHASES AND DETAILS

To empirically investigate the accuracy and efficiency of our proposed methods, we divide the experiments into **at most** 4 distinct phases including both federated and personalized learning stages, similarly to [32]. We show in Section V-D that not all phases are necessary and that their adoption mainly depends on the dataset's spit heterogeneity level and the available communication and computational budget. The experimental phases are defined as follows.

Phase 1: *Classifier initialization (ψ init).*

The classifier weights θ_ψ are initialized with `Fed3R` weights W (or `Fed3R-RF` in Appendix A, or `FedNCM` in Appendix D). In case the classifier is not initialized (*i.e.*, it is randomly initialized), this is denoted by ✗.

Phase 2: *Federated Fine-Tuning (**FT**).*

This phase corresponds to standard federated training. We refer to this phase as *fine-tuning* because it always starts from a pre-trained initialization on a general-purpose dataset (*e.g.*, ImageNet [96]), and it could fine-tune, or not, the `Fed3R` classifier.

- We run experiments with `FedAvg` [1], `FedAvgM` [35], `Scaffold` [6], `Mime` [38], and `FedDyn` [7]. However, despite our best efforts in hyper-parameters tuning, `Mime` and `FedDyn` failed to converge in both the Google Landmarks and iNaturalist experiments. Additional details are shown in Appendix D.
- By default, we perform training for 3000 rounds in all the scenarios except for iNaturalist, for which we train the model for 5K rounds since more rounds were necessary for baseline algorithms to reach convergence on the iNaturalist-User-120K split. Otherwise, we specify the number of rounds by adding `-num_rounds` at the end of the algorithm name (*e.g.*, `FedAvg-1k`).
- If not explicitly indicated, or if specified with the notation `FT(`$\mathcal{M}$`)`, we fine-tune the parameters of the whole model. However, as described in IV-B, it is also possible to only fine-tune the parameters of the classifier or the parameters of the feature extractor. In these cases, we use the notations `FT(`φ`)` and `FT(`ψ`)`, respectively.

Phase 3: *Partial Personalization (**PP**).*

This phase involves partial personalization training, as described in Section III-C, where a subset of the model parameters is shared among clients, while the remaining parameters are kept private (personal). We conduct experiments with three established PFL algorithms: `FedSim` [32], `FedAlt` [32], and `pFedMe` [73], running each for 1000 rounds. Further details on the experimental setup and hyperparameter choices are provided in Appendix D.

To investigate the interaction between statistical heterogeneity and personalization on different model components, we focus on the classifier and feature extractor and consider the following personalization strategies:

- Shared feature extractor parameters θ_φ, personal classifier parameters θ_ψ. This strategy is denoted as `Name_Alg(`ψ`)`.
- Shared classifier parameters (θ_ψ), personal feature extractor parameters θ_φ. This strategy is denoted as `Name_Alg(`φ`)`.

This focus is motivated by the observation that the classifier is generally more susceptible to bias in federated settings [30]. By comparing these two strategies, we aim to understand how the personalization of different components affects performance in the presence of statistical heterogeneity.

Phase 4: *Full Personalization (**FP**).*

This phase encompasses `FP` training, as introduced in Section III-C, where all, or a subset, of the model parameters are locally fine-tuned and there is no sharing across clients. In all the experiments that include this phase, all the clients are locally fine-tuned for 25 epochs.

- By default, the model is fine-tuned using the SGD optimizer. As an alternative, we add the `Ditto` [74] regularization term, which penalizes personalized updates that diverge from the global model.
- We indicate experiments performing `FP` on all the model parameters with `FP(`$\mathcal{M}$`)`. Instead, we denote by `FP(`φ`)` and `FP(`ψ`)` the experiments that fine-tune only the parameters of the feature extractor or of the classifier, respectively.

Aggregate notation.

Since the experiments may include at most four phases, explicitly indicating all four phases in the text can be somewhat difficult and redundant. Therefore, instead of describing a particular experiment as one where the classifier is initialized with `Fed3R` parameters, fine-tuned with `FedAvg`, using `FedSim` as the partial personalization strategy and `FT(`$\mathcal{M}$`)` as the full personalization strategy, we simply express it as `Fed3R + FedAvg + FedSim + FT(`$\mathcal{M}$`)`.

Additionally, to indicate that the columns of the classifier parameters have been pruned using `OLL`, we use the notation `OLL(`$\cdot$`)`, where $\cdot$ can represent any combination of strategies. For example, to specify a strategy that includes `Fed3R` in Phase 1 and `FedAvg` in Phase 2, followed by pruning the classifier parameters with `OLL` and finally fully personalizing the model of each client on the local distribution, we write it as `OLL(Fed3R + FedAvg) + FT(`$\mathcal{M}$`)`.

B. DATASETS AND EVALUATION

In this work, we focus on image classification datasets for FL and PFL. For both FL and PFL scenarios, we perform our experiments on two large-scale datasets: Google Landmarks (2028 classes) [27] and iNaturalist (1203 classes) [28].

Federated Learning splits

For the FL experiments, we use the splits provided in [29]. Specifically, for Google Landmarks, we perform our experiments on the only available split, the Users-160K split, while for iNaturalist, we perform the experiments on the Users-160K, Geo-1K and Geo-3K splits. The Users-160K split is the split that provides the largest number of clients (∼10K clients, ten times more than Google Landmarks-User160K, which counts ∼1K clients). Additional experiments using the Cifar100 dataset [97] are presented in Appendix H.

Personalized Federated Learning Splits

For our PFL experiments, we use a modified version of the User-160K split of Google Landmarks, as proposed by [32], which we name Pers-User-160K for convenience. This split ignores the clients with less than 50 images in the original User-160K split and the original global test set of the Users-160K split. Instead, it divides each client into a local training and a test dataset, assigning about 45% of the total local samples to the test dataset. Pers-User-160K counts a total of ∼800 clients. Similarly, in this work, we construct a novel Pers-Geo-100 split for the iNaturalist, starting from the Geo-100 split proposed in [29] and following a similar protocol to [32]. However, we keep all the clients with 10 or more local images (instead of 50) to simulate a more heterogeneous and challenging scenario. We preferred Geo-100 split to Users-120K because the large majority of the clients in Users-120K count less than 10 images. Our proposed Pers-Geo-100 split, results in a total of ∼2700 clients.

Measures of heterogeneity

Figure 5 shows the different levels of heterogeneity of the iNaturalist and Google Landmarks splits, comparing the proportion of the clients per class, the proportion of the classes per client, and the Mean Jaccard Index [98], [99] computed across the class distributions of all the couples of clients as follows:

$$\bar{J} = \frac{1}{K^2} \sum_{k \in \mathcal{K}} \sum_{h \in \mathcal{K}} \frac{\mathcal{C}_k \cap \mathcal{C}_h}{\mathcal{C}_k \cup \mathcal{C}_h}. \tag{9}$$

The Mean Jaccard Index $\bar{J}$ quantifies the statistical heterogeneity by measuring the overlap between the class distributions of different client pairs. A higher $\bar{J}$ indicates a greater similarity between the client pairs'class distributions, while a lower value suggests more statistical heterogeneity.

Evaluation protocol

For the evaluation, in the FL setting, we assume that there exists an external dataset $\mathcal{D}_{test}$, inaccessible during training, representative of all the local data distributions of the clients $k \in \mathcal{K}$. Instead, in the PFL setting, we assume that each client $k \in \mathcal{K}$ possesses an additional dataset $\mathcal{D}_k^{test}$, only for evaluation, representative of the local data distribution of client k. Therefore, we evaluate the performance of the FL experiments using the accuracy of the test client while we evaluate the performance of the PFL experiments using the balanced accuracy over the test datasets $\mathcal{D}_k^{test}$.

Additional details on the datasets are provided in Appendix D. In the following, we refer to the Google Landmarks and iNaturalist datasets without specifying the specific splits, meaning the Google Landmarks-User-160K and iNaturalist-User-120K splits for FL, and the Google Landmarks-Pers-User-160K and iNaturalist-Pers-Geo-100 splits for PFL. We specify the splits only for a minority of the experiments presented in Appendix V-C.

C. FEDERATED LEARNING EXPERIMENTS

In this Section, we present the experiments in the classical FL setting, which only include the *classifier initialization* and *fine-tuning* phases introduced in Section V-A. First, we empirically show that `Fed3R` is immune to the statistical heterogeneity. Then, we compare the new version of the `Fed3R` algorithm with the one from [31], showing how communicating only the vectors b_k^c drastically reduces the required communication. Finally, we analyze the benefits of the initialization of the classifier parameters with the `Fed3R` parameters, showing that this strategy accelerates the convergence of FL algorithms, mitigating the negative impact of statistical heterogeneity.

`Fed3R` immunity to statistical heterogeneity

In Figure 6, we compare `Fed3R` with `FedAvg(`ψ`)` on four different iNaturalist splits with different levels of statistical heterogeneity. In these experiments, for visualization purposes, we simulate one round for `Fed3R` as $\kappa = 10$ clients sharing the updates with the server and the server updating its solution. Notably, despite the differences in the number of clients and data distribution across the clients, `Fed3R` always achieves the same accuracy of 45.1% independently from the specific split, confirming that `Fed3R` is immune to statistical heterogeneity, as discussed in Section IV-A. On the other hand, `FedAvg(`ψ`)` is much slower, noisier, and less performative than `Fed3R`, especially in the most heterogeneous User-120K split, highlighting how `FedAvg(`ψ`)` is negatively affected from statistical heterogeneity. Moreover, conversely to gradient-based FL methods, `Fed3R` with the simulation of the rounds is guaranteed to always achieve maximum accuracy in $\lceil K/\kappa \rceil$ rounds. Consequently, a higher participation rate means faster convergence of `Fed3R`. This is not guaranteed, in general, for gradient-based FL algorithms.

Comparison with [31] version of `Fed3R`

In Figure 7, we empirically show that the new version of `Fed3R` is, on average, almost 10 times less communication expensive than the [31] version in both the splits. This is thanks to the fact that, in the new version, each client has to

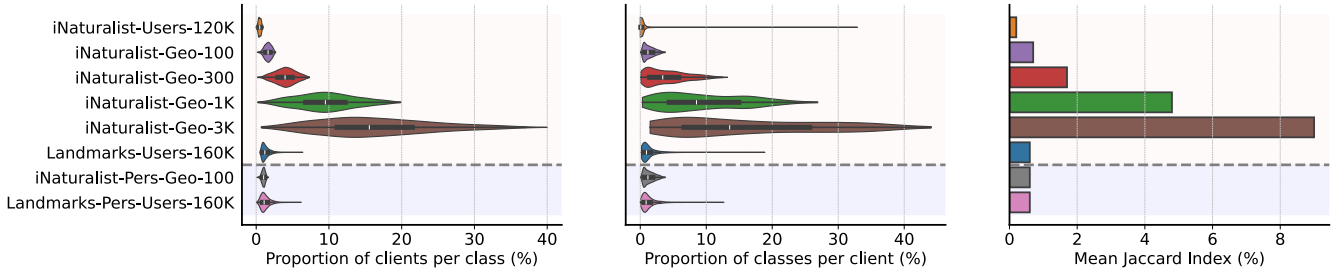

FIGURE 5. Statistical heterogeneity for the splits of iNaturalist and Google Landmarks, using three different metrics. On the left: distribution of the proportion of clients per class (relative to the total number of classes). In the middle: distribution of the proportion of classes per client (relative to the total number of clients). On the right: The Mean Jaccard Index. All values are presented as percentages. The splits above the dashed line represent FL splits, while those below are PFL splits.

TABLE 2. Comparison of `Fed3R` with `FedNCM` [52], another closed-form classifier existing in literature, and other gradient-based FL algorithms at convergence, without classifier parameters initialization. ψ indicates the classifier, `Name_Alg(`ψ`)` indicates federated fine-tuning of the classifier only while keeping the pre-trained feature extractor parameters fixed, and `Name_Alg` indicates federated fine-tuning of the whole model. We use the N/A notation for the results of the experiments that failed to converge.

(a) iNaturalist User-120K

Algorithm	Accuracy (%)	Total communication	Avg. computation per client
`Fed3R`	**45.1** ± **0.0**	**30.7 GB**	4.0 GFLOPs
`FedNCM`	32.2 ± 0.0	57.1 GB	**3.2 GFLOPs**
`FedAvg(`ψ`)`	36.7 ± 0.4	616.9 GB	108.6 GFLOPs
`FedAvgM(`ψ`)`	37.6 ± 0.2	616.9 GB	108.6 GFLOPs
`Scaffold(`ψ`)`	N/A	N/A	N/A
`FedAvg`	39.5 ± 3.2	1.5 TB	322.9 GFLOPs
`FedAvgM`	39.3 ± 0.7	1.5 TB	322.9 GFLOPs
`Scaffold`	N/A	N/A	N/A

(b) iNaturalist Geo-1K

Algorithm	Accuracy (%)	Total communication	Avg. computation per client
`Fed3R`	45.1 ± 0.0	**1.4 GB**	**108.9 GFLOPs**
`FedNCM`	32.2 ± 0.0	2.3 GB	**108.5 GFLOPs**
`FedAvg(`ψ`)`	44.1 ± 0.5	124.0 GB	550.2 GFLOPs
`FedAvgM(`ψ`)`	43.9 ± 0.3	124.0 GB	550.2 GFLOPs
`Scaffold(`ψ`)`	47.1 ± 0.4	248.0 GB	550.2 GFLOPs
`FedAvg`	51.0 ± 0.1	296.0 GB	1.6 TFLOPs
`FedAvgM`	51.1 ± 0.4	296.0 GB	1.6 TFLOPs
`Scaffold`	**52.5** ± **0.2**	592.0 GB	1.6 TFLOPs

(c) iNaturalist Geo-3K

Algorithm	Accuracy (%)	Total communication	Avg. computation per client
`Fed3R`	45.1 ± 0.0	**579.6 MB**	**297.6 GFLOPs**
`FedNCM`	32.2 ± 0.0	831.5 MB	**296.7 GFLOPs**
`FedAvg(`ψ`)`	46.0 ± 0.9	124.0 GB	1.5 TFLOPs
`FedAvgM(`ψ`)`	46.9 ± 0.3	124.0 GB	1.5 TFLOPs
`Scaffold(`ψ`)`	48.0 ± 0.3	248.0 GB	1.5 TFLOPs
`FedAvg`	47.8 ± 3.5	296.0 GB	4.5 TFLOPs
`FedAvgM`	**50.8** ± **1.2**	296.0 GB	4.5 TFLOPs
`Scaffold`	46.4 ± 3.6	592.0 GB	4.5 TFLOPs

(d) Google Landmarks

Algorithm	Accuracy (%)	Total communication	Avg. computation per client
`Fed3R`	49.6 ± 0.0	**4.4 GB**	40.0 GFLOPs
`FedNCM`	36.2 ± 0.0	13.1 GB	**31.7 GFLOPs**
`FedAvg(`ψ`)`	41.0 ± 1.6	618.8 GB	4.8 TFLOPs
`FedAvgM(`ψ`)`	40.8 ± 1.2	618.8 GB	4.8 TFLOPs
`Scaffold(`ψ`)`	51.7 ± 0.3	1.2 TB	4.8 TFLOPs
`FedAvg`	57.7 ± 1.2	1.1 TB	14.1 TFLOPs
`FedAvgM`	58.7 ± 0.8	1.1 TB	14.1 TFLOPs
`Scaffold`	**63.4** ± **0.9**	2.3 TB	14.1 TFLOPs

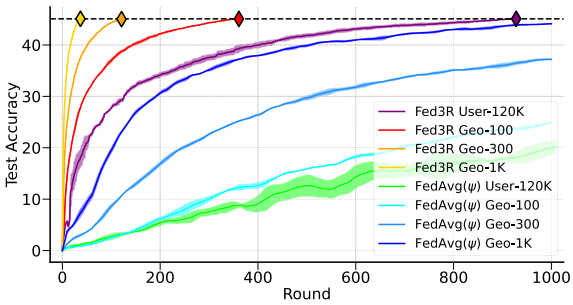

FIGURE 6. Empirical demonstration of the immunity to statistical heterogeneity of the `Fed3R` algorithm. This plot simulates rounds for the `Fed3R` algorithm, comparing `Fed3R` with `FedAvg(`ψ`)` on different splits of iNaturalist, with groups of $\kappa = 10$ clients, on different levels of statistical heterogeneity (from the split with the lowest statistical heterogeneity to the highest: Geo-1K, Geo-300, Geo-100, Users-120K). Details on these splits are presented in Table 8.

communicate only the $A_k \in \mathbb{R}^{d\times d}$ matrix and the $b_k^c \in \mathbb{R}^d$ vectors associated with the local classes, compared with the ICML version of the algorithm that communicates the whole matrix $b_k \in \mathbb{R}^{d\times C}$ along with the matrix A_k.

Comparison between `Fed3R` and FL baselines

Table 2 compares `Fed3R` with other FL algorithms. In particular, we show the accuracy, the communication and computation costs of several gradient-based FL methods. Additionally, we also provide the same metrics for `FedNCM`.

`Fed3R` and `FedNCM` are consistently 100 to 1000 times more efficient in terms of computations and communication than the gradient-based baseline methods. However, **`Fed3R` outperforms `FedNCM` by approximately 13 percentage points in accuracy on both the Google Landmarks and iNaturalist datasets, while requiring similar communication and computation costs.**

Additionally, despite `Fed3R` relying solely on the expressive power of the pre-trained feature extractor for the target FL task (*i.e.*, the feature extractor remains fixed and

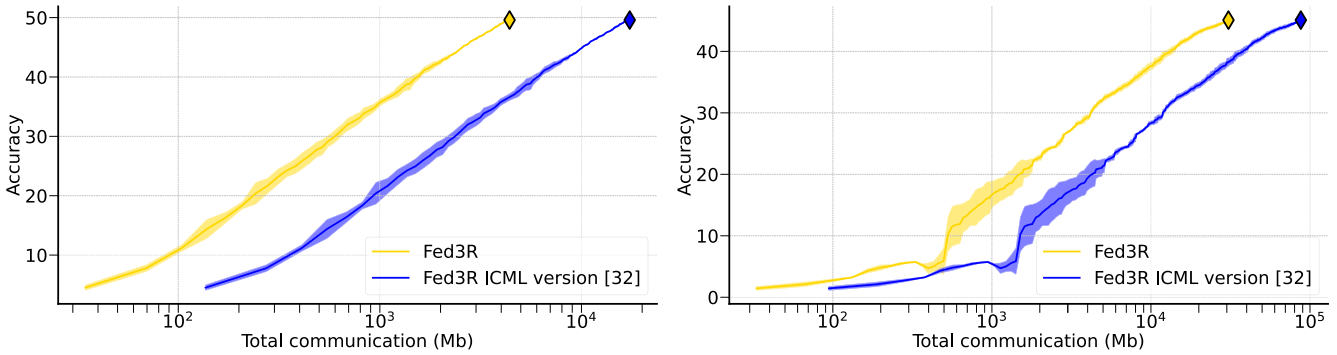

FIGURE 7. Comparison between the total communication costs of the new version of the Fed3R algorithm with its ICML version [31]. On the left, we show the communication costs for Google Landmarks, while we show the communication costs for iNaturalist on the right.

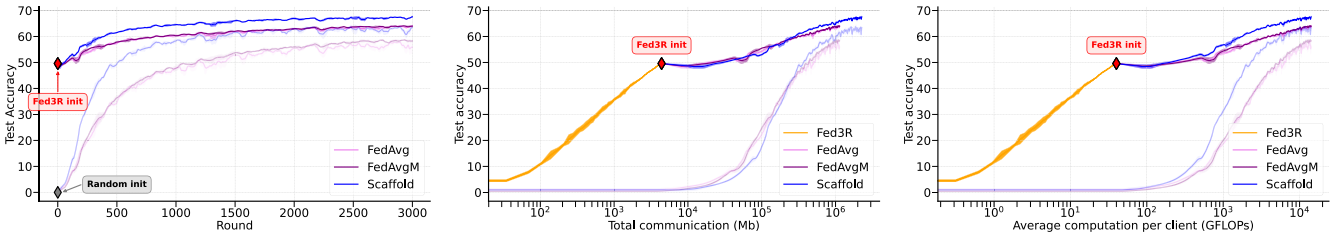

FIGURE 8. Comparison between FL algorithms on the Google Landmarks dataset, and the same algorithms fine-tuned after initializing the classifier parameters using Fed3R. From left to right: test accuracy by rounds, test accuracy by total communication budget, test accuracy by expected computation per client, on average. The red marker indicates the Fed3R initialization point, which allows fine-tuning with any FL algorithm. Less intense colors are associated with the baseline algorithms without Fed3R initialization.

Fed3R only constructs a classifier), it achieves the highest accuracy on iNaturalist-Users-120K compared to the other methods. This holds even when compared with FedAvg and FedAvgM, which instead fine-tune the entire model, including the feature extractor. This result emphasizes the effectiveness of our classifier initialization in real-world, cross-device settings with high statistical heterogeneity.

Effects of the Fed3R initialization

As already shown, Fed3R is already sufficient to surpass baseline methods on highly statistically heterogeneous scenarios such as iNaturalist-Users-120K. However, in the iNaturalist-Geo-3K, iNaturalist-Geo-1K and Google Landmarks experiments, which are less affected by statistical heterogeneity, Fed3R alone is not sufficient to surpass the FL baselines, although its accuracy is still higher in scenarios with a limited budget of communication or computational costs. Therefore, as explained in Section IV-B, and motivated by the aim of breaking the vicious cycle presented in [17], we propose using the optimal Fed3R parameters W^* to initialize the classifier's parameters θ_ψ, and then fine-tuning the model with an FL algorithm of choice.

Figure 8 shows that this strategy improves accuracy compared with the same algorithms without Fed3R initialization while retaining computational and communication gains. Moreover, it stabilizes the training, accelerating convergence.

Fine-tuning the entire model from the Fed3R classifier initialization is effective in mildly heterogeneous settings like Google Landmarks, or in relatively low heterogeneous settings like iNaturalist-Geo-1K and iNaturalist-Geo-3K. However, as shown in Figure 9, this approach (Fed3R + FT($\mathcal{M}$)) is not appropriate for more heterogeneous datasets such as iNaturalist-Users-120K, which has 10 times the number of clients of Google Landmarks, more than 25 times the number of clients of iNaturalist-Geo-1K and about 69 times the number of clients of iNaturalist-Geo-3K. Indeed, fine-tuning the entire model can reduce predictive performance and convergence speed due to classifier bias, causing destructive interference during the aggregation phase, as discussed in Section I. Figure 9 shows that accuracy does not improve when only the classifier is fine-tuned (Fed3R + FT(ψ) experiment), as the feature extractor is not optimized for the target task. This demonstrates that the pre-trained features are not expressive enough for the target task. On the other hand, keeping the classifier fixed and fine-tuning only the feature extractor effectively prevents the issues caused by heterogeneity, thus avoiding interference and allowing the feature extractor to adapt to the Fed3R classifier and the target task, improving final accuracy.

Finally, Table 3 summarizes the results for both Google Landmarks and iNaturalist experiments, highlighting the consistent benefits of initializing the classifier with Fed3R parameters. Across all scenarios, fine-tuning with Fed3R initialization leads to significantly higher final accuracies compared to training from scratch.

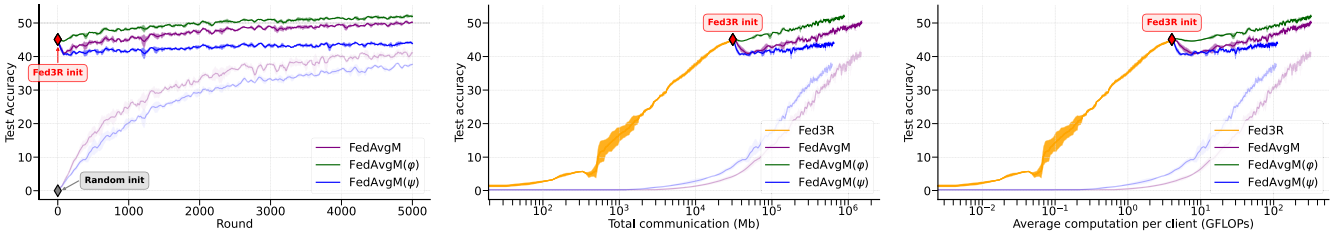

FIGURE 9. Comparison between different FT strategies on the iNaturalist-Users-120K dataset. We show results using FedAvgM as FT algorithm, as the results with FedAvg were similar but noisier. We omit the FedAvgM(φ) experiment without Fed3R initialization because this experiment would achieve poor results as the classifier is randomly initialized and not fine-tuned. From left to right: test accuracy by rounds, test accuracy by total communication budget, test accuracy by expected computation per client, on average. The red marker indicates the Fed3R initialization point. Less intense colors are associated with the baseline algorithms without Fed3R initialization.

TABLE 3. Comparison between fine-tuning strategies, with and without Fed3R initialization, using different FL baselines as FT algorithms (Acc. (%)). FT(φ) indicates federated fine-tuning of the feature extractor, keeping the classifier parameters fixed; FT(ψ) indicates federated fine-tuning of the classifier keeping the feature extractor parameters fixed; FT($\mathcal{M}$) indicates federated fine-tuning of the whole model parameters. Since when the classifier ψ is not initialized with the Fed3R parameters, it is randomly initialized, the performance of the FT(φ) strategy would always be near 0. Therefore, they are omitted (indicated with the - symbol). Scaffold failed to converge in all the experiments in the iNaturalist-Users-120K scenario (indicated with N/A).

(a) **iNaturalist Users-120K**

ψ init	FT alg	FT(φ)	FT(ψ)	FT($\mathcal{M}$)
✗	FedAvg	-	36.7 ± 0.4	39.5 ± 3.2
Fed3R	FedAvg	50.8 ± 0.2	42.0 ± 0.3	49.0 ± 0.6
✗	FedAvgM	-	37.6 ± 0.2	39.3 ± 0.7
Fed3R	FedAvgM	**51.5 ± 0.2**	43.5 ± 1.9	49.8 ± 0.8
✗	Scaffold	-	N/A	N/A
Fed3R	Scaffold	N/A	N/A	N/A

(b) **iNaturalist Geo-1K**

ψ init	FT alg	FT(φ)	FT(ψ)	FT($\mathcal{M}$)
✗	FedAvg	-	44.1 ± 0.5	51.0 ± 0.1
Fed3R	FedAvg	53.7 ± 0.2	49.2 ± 0.4	53.6 ± 0.1
✗	FedAvgM	-	43.9 ± 0.3	51.1 ± 0.4
Fed3R	FedAvgM	53.8 ± 0.1	49.4 ± 0.2	53.3 ± 0.2
✗	Scaffold	-	47.1 ± 0.4	52.5 ± 0.2
Fed3R	Scaffold	53.7 ± 0.1	49.3 ± 0.2	**55.7 ± 0.1**

(c) **iNaturalist Geo-3K**

ψ init	FT alg	FT(φ)	FT(ψ)	FT($\mathcal{M}$)
✗	FedAvg	-	46.0 ± 0.9	47.8 ± 3.5
Fed3R	FedAvg	50.8 ± 1.6	47.0 ± 1.4	50.9 ± 1.5
✗	FedAvgM	-	46.9 ± 0.3	50.8 ± 1.2
Fed3R	FedAvgM	50.1 ± 0.2	46.4 ± 0.3	51.0 ± 0.2
✗	Scaffold	-	48.0 ± 0.3	46.4 ± 3.6
Fed3R	Scaffold	49.3 ± 2.3	47.0 ± 1.2	**54.1 ± 1.7**

(d) **Google Landmarks**

ψ init	FT alg	FT(φ)	FT(ψ)	FT($\mathcal{M}$)
✗	FedAvg	-	41.0 ± 1.6	57.7 ± 1.2
Fed3R	FedAvg	59.6 ± 0.2	56.7 ± 0.4	64.1 ± 0.4
✗	FedAvgM	-	40.8 ± 1.2	58.7 ± 0.8
Fed3R	FedAvgM	59.0 ± 0.2	56.2 ± 0.5	64.1 ± 0.2
✗	Scaffold	-	51.7 ± 0.3	63.4 ± 0.9
Fed3R	Scaffold	63.4 ± 0.1	58.0 ± 1.5	**67.4 ± 0.3**

Moreover, as illustrated in Figures 8 and 9, our methods consistently outperform the baselines in terms of accuracy, communication efficiency, and computational cost. These results underscore the critical role of robust initialization in FL and demonstrate how Fed3R effectively fulfills this role. By providing optimal classifier parameters equivalent to the centralized RR solution, Fed3R ensures immunity to statistical heterogeneity and facilitates efficient and effective learning in challenging federated environments.

In Appendix I, we show that Fed3R is robust to domain shift and can be combined with other modular techniques to further improve accuracy.

D. PERSONALIZED FEDERATED LEARNING EXPERIMENTS

In this Section, we present the PFL experiments, which comprise all the four phases introduced in Section V-A. First, we present and discuss the results without Phase 4 (FP), *i.e.*, the results of the experiments with or without Fed3R initialization, FL training, and partial personalization. Then, we describe the results of the experiments with Phase 4. The objective is to understand the best PFL training strategies depending on the heterogeneity of the setting and the particular strategy adopted for all four phases.

1) Partial Personalization

In this Section, we examine the impact the initialization of the classifier with the Fed3R parameters and the federated fine-tuning phase have in the PFL setting for the PP strategy. Specifically, we demonstrate how the initialization of the classifier parameters using Fed3R optimal parameters W^* yields benefits comparable to those shown in Section V-C for the FL setting. Since pFedMe consistently provides lower accuracy compared to FedSim and FedAlt, which always perform similarly in all our experiments, here we focus solely on the differences between FedSim(φ) and FedSim(ψ), to emphasize the difference between the two PP strategies. The complete results for the experiments of this Section, including the omitted FedAlt and pFedMe results, are presented in Appendix Section VI.

TABLE 4. Comparison among personal and shared parameters selection for the PP experiments using FedSim as PP algorithm. ψ indicates the classifier; FedSim(ψ) indicates training with FedSim, with the classifier parameters as the personal parameters and the feature extractor parameters as the shared parameters; FedSim(φ) indicates training with FedSim, with the feature extractor parameters as the personal parameters and the classifier parameters as the shared parameters.

Dataset	ψ init	FT alg	FT acc	FedSim(ψ)	FedSim(φ)
iNaturalist	✗	FedAvg	46.3 ± 0.4	70.9 ± 0.2	67.4 ± 0.2
	✗	FedAvg-1k	30.0 ± 0.3	40.7 ± 0.1	52.1 ± 0.1
	Fed3R	✗	58.0 ± 0.0	75.1 ± 0.0	72.9 ± 0.1
	Fed3R	FedAvg	67.3 ± 0.2	**78.5 ± 0.1**	76.9 ± 0.1
	Fed3R	FedAvg-1k	63.5 ± 0.1	77.3 ± 0.1	75.5 ± 0.1
	Fed3R	FedAvg(φ)	63.1 ± 0.1	77.5 ± 0.0	77.2 ± 0.1
Google Landmarks	✗	FedAvg	61.2 ± 1.0	79.4 ± 0.0	79.7 ± 0.1
	✗	FedAvg-1k	47.6 ± 1.0	66.5 ± 0.2	74.1 ± 0.0
	Fed3R	✗	49.9 ± 0.0	72.9 ± 0.1	74.1 ± 0.1
	Fed3R	FedAvg	68.1 ± 0.4	79.9 ± 0.1	**80.4 ± 0.0**
	Fed3R	FedAvg-1k	63.0 ± 0.3	77.0 ± 0.3	78.6 ± 0.1
	Fed3R	FedAvg(φ)	68.2 ± 0.1	79.9 ± 0.1	**80.4 ± 0.0**

Table 4 shows the weighted accuracy after the PP phase with FedSim(ψ) of different strategies for Phases 1 and 2, on both the Google Landmarks and iNaturalist dataset splits.

PP improves the final accuracy of the FT experiments. Overall, PP significantly improves the original accuracy. For instance, the weighted accuracy of the FedAvg experiment without the Fed3R initialization on Google Landmarks increases from 61.2% to 79.7% after the PP phase using FedSim(φ).

Comparison between PP(φ) and PP(ψ) strategies. On Google Landmarks, FedSim(φ) consistently outperforms FedSim(ψ). Conversely, on most of the iNaturalist experiments, FedSim(ψ) oupterforms FedSim(φ), with the sole exception of the FedAvg-1k experiment without Fed3R initialization, where FedSim(φ) accuracy is 12% points above the accuracy of FedSim(ψ). This discrepancy between Google Landmarks and iNaturalist results is due to the data recency bias problem, which causes the local classifiers of the clients to drift apart, as they are optimized for the local distribution. Indeed, on the more challenging and heterogeneous iNaturalist split, keeping the classifier local while aggregating the parameters of the feature extractor helps prevent destructive interference on the classifier while fine-tuning the feature extractor globally. On the other hand, in the less heterogeneous Google Landmarks scenario, specializing the feature extractor on the target task while keeping the classifier local effectively improves the quality of the feature extractor, which can benefit from knowledge extracted from all the clients.

PP(ψ) is robust to the quality of the FT classifier. Our experiments reveal an interesting trend in the PP phase. We observe that the performance difference between initializing with a fully fine-tuned model (FT($\mathcal{M}$)) and initializing with a fine-tuned feature extractor only (FT(φ)) becomes negligible. For instance, on iNaturalist, the weighted accuracy difference between Fed3R + FedAvg + FedSim(ψ) and Fed3R + FedAvg(φ) + FedSim(ψ) is only 1%, whereas the difference between Fed3R + FedAvg and Fed3R + FedAvg(φ) (without PP) was almost 4%.

This suggests that when using PP methods with the classifier parameters as local parameters (FedSim(ψ)), the local classifiers can effectively specialize regardless of the initial classifier state. Whether the initial classifier is the robust Fed3R classifier or a fine-tuned version potentially affected by data recency bias and destructive interference appears to have minimal impact on the final personalized performance. This finding highlights the ability of PFL methods to overcome the limitations of previous training phases and achieve effective personalization.

When statistical heterogeneity is high, Fed3R initialization makes FT Phase unnecessary. Results in Table 4 show that in highly heterogeneous settings such as iNaturalist, the FT phase can be avoided if the classifier is initialized with Fed3R parameters. Indeed, FedSim(ψ) with Fed3R initialization and no fine-tuning achieves 75.1% accuracy, compared with the same PP algorithm but with FedAvg as FT algorithm and no Fed3R initialization, which reaches only 70.9%. This result underscores that only 1K rounds of PP with Fed3R initialization are sufficient to surpass the performance with 3K rounds of FT and 1K rounds of PP, therefore saving 75% of the total training rounds. Additionally, with enough time and resources available, the full pipeline with Fed3R initialization, FT Phase, and PP Phase is the one that provides the best results, higher than the results of FT + PP without Fed3R initialization, in both the settings. Once again, these results stress the importance of having a robust classifier initialization using Fed3R in scenarios with high statistical heterogeneity.

Fed3R initialization reduces the necessary rounds for the FT Phase in mildly heterogeneous scenarios. Google Landmarks experiments do not exhibit the same desirable results exposed in the previous paragraph for iNaturalist, where Fed3R initialization allows avoiding the federated FT phase at all. Nevertheless, Figure 10 shows that running the FT phase for only 1K rounds starting from Fed3R initialization and then performing the PP phase is sufficient to achieve performances comparable to the best results that can be obtained with the same strategy but with 3K rounds in the FT steps, therefore saving two-thirds of the FT rounds and, consequently, communication and computational costs. Specifically, Fed3R + FedAvg-1k + FedSim(φ) reaches 78.6% accuracy, only 1 point below FedAvg + FedSim(φ) without Fed3R initialization and 2 points below Fed3R + FedAvg + FedSim(φ), but saving 2K rounds compared to these two strategies.

2) Full Personalization

This section analyzes the results of FP experiments, and presents several insights, focusing on the impact of Fed3R initialization and the trade-offs of different intermediate phases. Complete results, including comparisons with the Ditto baseline and various combinations of PP and FP phases, are presented in Appendix F. We omit the

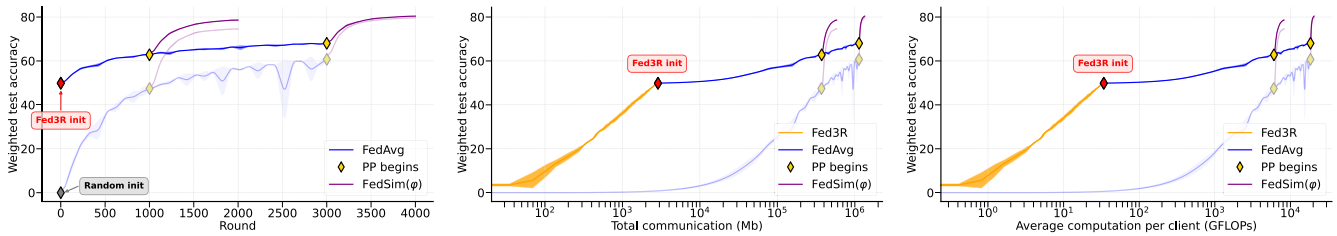

FIGURE 10. From left to right: Partial Personalization performance, communication, and computation costs with and without `Fed3R` classifier initialization, and with 1K or 3K rounds for the `FT` phase (`FedAvg`), in the Google Landmarks scenario. For this visualization, we choose the `FedSim(`φ`)` algorithm for the `PP` phase. Less intense colors are associated with the baseline algorithms without `Fed3R` initialization.

TABLE 5. Comparison between `FP` strategies with different choices for the initialization, `FT`, and `PP` phase, on both iNaturalist and Google Landmarks. ψ indicates the classifier; φ indicates the feature extractor; $\mathcal{M}$ indicates the whole model; `FP(`$\cdot$`)` indicates full personalization of the parameters associated with $\cdot$, while the other parameters (if any) are fixed.

ψ init	FT alg	PP alg	iNaturalist				Google Landmarks			
			✗	FP(ψ)	FP(φ)	FP($\mathcal{M}$)	✗	FP(ψ)	FP(φ)	FP($\mathcal{M}$)
Fed3R	✗	✗	58.0 ± 0.0	77.4 ± 0.1	78.6 ± 0.2	77.1 ± 0.1	49.9 ± 0.0	68.4 ± 0.2	68.9 ± 0.1	69.7 ± 0.1
✗	FedAvg	✗	46.3 ± 0.4	75.2 ± 0.1	75.5 ± 0.2	78.1 ± 0.1	61.2 ± 1.0	78.3 ± 0.1	78.3 ± 0.2	80.0 ± 0.0
Fed3R	FedAvg	✗	67.3 ± 0.2	83.8 ± 0.0	84.1 ± 0.2	83.3 ± 0.0	68.1 ± 0.4	80.0 ± 0.0	79.5 ± 0.1	80.0 ± 0.1
Fed3R	✗	FedSim(φ)	72.9 ± 0.1	83.4 ± 0.1	84.8 ± 0.1	82.8 ± 0.1	74.1 ± 0.1	73.1 ± 0.2	74.5 ± 0.1	73.0 ± 0.1
✗	FedAvg	FedSim(φ)	67.4 ± 0.2	82.1 ± 0.1	79.3 ± 0.1	82.4 ± 0.1	79.7 ± 0.1	80.2 ± 0.0	80.0 ± 0.0	80.3 ± 0.1
Fed3R	FedAvg	FedSim(φ)	76.9 ± 0.1	86.8 ± 0.1	**87.4 ± 0.1**	86.4 ± 0.1	80.4 ± 0.0	80.3 ± 0.1	**80.7 ± 0.1**	80.2 ± 0.1

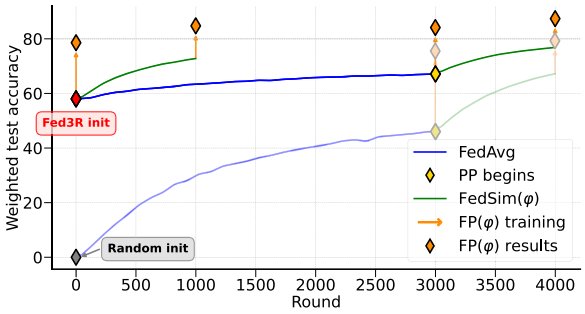

FIGURE 11. Comparison of Full Personalization results after different possibilities for the first three phases, for the iNaturalist setting, using `FedSim(`φ`)` and `FP(`φ`)`. Less intense colors are associated with the baseline algorithms without `Fed3R` initialization.

`Ditto` experiments from this section for clarity, as they generally show negligible improvements over standard `SGD` optimization. Table 5 summarizes the `FP` experiments, with `FedSim(`φ`)` selected as the `PP` algorithm due to its slight advantage in weighted accuracy compared to other `PP` strategies like `FedSim(`ψ`)`, `FedAlt(`φ`)`, and `FedAlt(`ψ`)`, as detailed in Appendix F.

Robust `Fed3R` initialization improves final `FP` accuracy. The results in Table 5 clearly demonstrate the positive impact of `Fed3R` initialization on `FP` performance. Across both the Google Landmarks and iNaturalist scenarios, all the experiments with `Fed3R` initialization consistently achieve higher accuracy than their counterpart without `Fed3R` initialization. For example, `Fed3R` + `FedAvg` + `FedSim(`φ`)` + `FT(`φ`)` achieves 87.4% accuracy on iNaturalist and 80.7% accuracy on Google Landmarks, compared with `FedAvg` + `FedSim(`φ`)` + `FT(`φ`)` that achieves only 79.3% on iNaturalist and 80.0% on Google Landmarks. Similarly, `Fed3R` + `FedAvg` + `FT(`φ`)` achieves 84.1% accuracy on iNaturalist and 79.5% accuracy on Google Landmarks, compared with `FedAvg` + `FT(`φ`)` that achieves only 75.5% on iNaturalist and 78.3% on Google Landmarks.

`PP` and `FP` phases are not both necessary in mildly heterogeneous settings. Interestingly, the necessity of combining `PP` and `FP` appears to depend on the level of data heterogeneity. In the mildly heterogeneous Google Landmarks scenario, the performance gains from combining both phases are minimal. For instance, `Fed3R` + `FedAvg` + `FedSim(`φ`)` + `FT(`φ`)` achieves 80.7% accuracy, only slightly higher than the 79.5% accuracy of `Fed3R` + `FedAvg` + `FT(`φ`)`. Conversely, on the more heterogeneous iNaturalist dataset, combining `PP` and `FP` consistently leads to significant improvements. This suggests that in highly heterogeneous settings, both forms of personalization are necessary to fully exploit the potential of personalized models. Given that the `PP` phase is generally more resource-intensive than `FP`, it may be more efficient to focus solely on `FP` in mildly heterogeneous scenarios like Google Landmarks as it enables comparable performance with reduced computational and communication overhead.

In highly heterogeneous scenarios, `Fed3R` initialization + `PP` + `FP` is the strategy with the highest performance-costs ratio. In the previous Section, we showed that `Fed3R` initialization allows avoiding the `FT` phase in highly statistically heterogeneous scenarios, saving 75% of the total rounds and costs. As Figure 11 shows for the iNaturalist dataset, this finding holds also when including the final `FP` phase In particular, `Fed3R` + `FedSim(`φ`)` + `FP(`φ`)` achieves only 2.6 percentage points of weighted

accuracy less than Fed3R + FedAvg + FedSim(φ) + FP(φ), **but saves 3000 rounds of training.** This substantial reduction in training rounds translates to significant savings in communication and computational costs, making Fed3R a highly efficient and scalable approach for achieving personalization in challenging federated settings. These results reinforce the value of Fed3R as a robust initialization that not only accelerates convergence but also enables substantial cost reduction by eliminating the need for expensive federated fine-tuning.

Fed3R initialization + FP final accuracy is comparable to the federated FT accuracy but drastically reduces the required communication. As Figure 11 and Table 5 show, the Fed3R initialization + FP(φ) strategy reaches 78.6% final weighted accuracy on iNaturalist, and the Fed3R initialization + FP($\mathcal{M}$) strategy achieves 69.7% final weighted accuracy on Google Landmarks, with no communication round required, necessitating only the limited communication costs of Fed3R. In contrast, federated FT without Fed3R initialization achieves only 46.3% and 61.2% on iNaturalist and Google Landmarks, respectively, while FT with Fed3R initialization reaches 67.3% on iNaturalist and 68.1% on Google Landmarks. Hence, Fed3R + FP demonstrates to be the best strategy when the allowed communication budget is particularly scarce.

E. ONLY LOCAL LABELS EXPERIMENTS

In this Section, we present the results of our OLL algorithm and the insights from our analysis. We show that OLL alone without further local fine-tuning of Phase 4 significantly improves the results by simply removing sources of potential classification errors from classes outside the local distribution. Moreover, we show how fine-tuning the OLL classifiers locally further improves the final weighted accuracy, as the new OLL classifier is streamlined for the local class distribution. Finally, we compare the effects of FP with and without OLL with a closer look at the clients.

Table 6 presents the final weighted accuracy of OLL applied to the classifier parameters at the end of the training of several Phases 1 and 2 strategies, comparing it with the best results without OLL among 4 possible algorithms for Phase 3. The column with the ✗ symbol for the results with OLL indicates that OLL is directly employed on the federated classifier from Phase 1 or Phase 2 to adapt it to each local clients' data distribution. In other words, we remove the columns associated with the classes in $\mathcal{C} \setminus \mathcal{C}_k$ from the latest classifier and then use it to perform predictions.

OLL consistently improves predictive performance. OLL improves the performance of any FP strategy. Notably, OLL without further fine-tuning in the FP phase already guarantees substantial performance improvements in both the Google Landmarks and iNaturalist scenarios, reaching a final weighted accuracy that is always near or above 80% for all the experiments. Remarkably, in the Google Landmarks dataset, OLL(Fed3R + FedAvg(φ)) reaches 84.0% accuracy, 3.3 percentage points of accuracy higher than the best strategy without OLL, *i.e.*, Fed3R + FedAvg + FedSim(φ) + FP(φ) which achieves only 80.7% accuracy. Moreover, OLL(Fed3R + FedAvg-1k) is also sufficient to improve results over the best method without OLL, achieving 81.1% accuracy and saving training time, communication, and computational costs. Notably, accuracy is even higher than the ones with FP($\mathcal{M}$) as the strategy for the FP phase. Indeed, OLL essentially imposes a strong bias towards locally present classes, effectively eliminating incorrect predictions for classes not represented locally — a result that the naïve FP strategies also achieve, but with more time and costs.

OLL dramatically simplifies the FP phases, further improving final accuracy. OLL significantly improves accuracy without the need for further local fine-tuning (we are referring to the results of OLL applied to any strategy including Phase 1 and/or Phase 2 and no Phase 4). However, in the iNaturalist scenario, strategies including Phase 3 and Phase 4 and without OLL still outperform the strategies with Phase 1, 2 and OLL. For instance, Table 5 shows that Fed3R + FedAvg + FP($\mathcal{M}$) achieves an accuracy of 83.3%, compared to 80.5% for OLL(Fed3R + FedAvg). Nevertheless, the simplified OLL classifier is designed to enhance subsequent FP optimization. Indeed, it prevents predictions for classes in $\mathcal{C} \setminus \mathcal{C}_k$, thereby eliminating many potential sources of incorrect predictions.

Notably, with OLL, we achieve a maximum accuracy of 92.5% on iNaturalist, compared to the best result of 88.1% without OLL, representing an improvement of 4.4%. On the Google Landmarks dataset, OLL yields a maximum accuracy of 85.5%, whereas the best result without OLL is 80.7%, corresponding to an improvement of 4.8%. Furthermore, the FedAvg-1k algorithm, when used as federated FT method with Fed3R initialization and OLL, achieves an accuracy of 85.4% on Google Landmarks and 86.4% on iNaturalist, saving two-thirds of the total FT rounds and thus reducing communication and computational costs.

FP inherently biases the classifier towards local classes, justifying the use of OLL. Finally, in Figure 12, we demonstrate that by using the FP strategy, the original classifier becomes increasingly biased towards local classes over time (see the orange vs blue curves). For static client distributions, as the one considered in this work, this is a positive outcome, as the goal is to maximize accuracy only on those local classes. In other words, we are not interested in the deterioration of the performance for classes outside the local distribution. While this is naturally happening with gradient-based FP, Figure 12 shows that it still requires several epochs to converge and reach the right balance. OLL dramatically accelerates this process by simplifying the local classification problem, explicitly disregarding classes outside the local distribution. As the personalized local classifier would be biased towards local classes anyway, this serves as a strong justification for OLL. By eliminating classes outside the local distribution, OLL leads to more efficient training by removing

TABLE 6. FP results after OLL, compared with the FP results without OLL, on both the Google Landmarks and iNaturalist datasets. ψ indicates the classifier; φ indicates the feature extractor; $\mathcal{M}$ indicates the whole model; FP $(\cdot)$ indicates full personalization of the parameters associated with $\cdot$, while the other parameters (if any) are fixed. The symbol † indicates that the displayed results are the best among the following 4 PP strategies (Phase 3): FedSim(ψ), FedSim(φ), FedAlt(ψ), and FedSim(φ). Conversely, the results using OLL do not include Phase 3, because OLL applied on local classifiers from Phase 1 and 2 is already sufficient to surpass the baselines by a large margin, even without Phase 3.

Dataset	ψ init	FT alg	With OLL				Without OLL			
			✗	FP (ψ)	FP (φ)	FP $(\mathcal{M})$	✗†	FP (ψ) †	FP (φ) †	FP $(\mathcal{M})$ †
iNaturalist	✗	FedAvg	76.9 ± 0.1	89.5 ± 0.1	88.5 ± 0.1	88.6 ± 0.2	71.0 ± 0.1	82.5 ± 0.2	81.8 ± 0.2	83.3 ± 0.1
	✗	FedAvg-1k	74.9 ± 0.1	76.6 ± 0.2	75.4 ± 0.1	72.7 ± 0.1	66.7 ± 0.3	79.8 ± 0.2	78.1 ± 0.1	81.1 ± 0.0
	Fed3R	✗	75.4 ± 0.0	88.6 ± 0.1	88.3 ± 0.1	87.9 ± 0.1	75.1 ± 0.0	85.6 ± 0.2	86.4 ± 0.1	85.3 ± 0.1
	Fed3R	FedAvg	80.5 ± 0.1	**92.5 ± 0.1**	92.3 ± 0.1	91.1 ± 0.1	78.5 ± 0.1	87.7 ± 0.1	88.1 ± 0.1	87.4 ± 0.1
	Fed3R	FedAvg-1k	78.9 ± 0.1	88.2 ± 0.0	88.5 ± 0.1	91.5 ± 0.1	77.3 ± 0.1	86.9 ± 0.1	87.4 ± 0.1	86.9 ± 0.1
	Fed3R	FedAvg (φ)	79.4 ± 0.2	86.4 ± 0.1	88.3 ± 0.1	88.5 ± 0.1	77.5 ± 0.0	87.0 ± 0.2	87.5 ± 0.1	86.4 ± 0.2
Google Landmarks	✗	FedAvg	82.6 ± 0.3	84.7 ± 0.1	84.0 ± 0.2	84.2 ± 0.1	79.7 ± 0.0	80.2 ± 0.0	80.4 ± 0.0	80.4 ± 0.0
	✗	FedAvg-1k	75.1 ± 0.3	79.2 ± 0.1	78.4 ± 0.2	77.8 ± 0.1	74.5 ± 0.1	75.3 ± 0.0	75.6 ± 0.1	75.6 ± 0.1
	Fed3R	✗	73.0 ± 0.0	75.7 ± 0.0	75.4 ± 0.1	75.5 ± 0.0	74.1 ± 0.1	73.1 ± 0.2	74.5 ± 0.1	73.5 ± 0.1
	Fed3R	FedAvg	83.9 ± 0.1	**85.5 ± 0.0**	85.4 ± 0.0	85.2 ± 0.1	80.4 ± 0.0	80.3 ± 0.1	80.7 ± 0.1	80.2 ± 0.1
	Fed3R	FedAvg-1k	81.1 ± 0.1	81.4 ± 0.1	81.5 ± 0.1	83.3 ± 0.1	78.6 ± 0.1	78.1 ± 0.0	78.9 ± 0.1	78.0 ± 0.1
	Fed3R	FedAvg (φ)	84.0 ± 0.0	84.0 ± 0.0	84.2 ± 0.1	85.4 ± 0.1	80.4 ± 0.0	80.3 ± 0.1	80.6 ± 0.1	80.1 ± 0.1

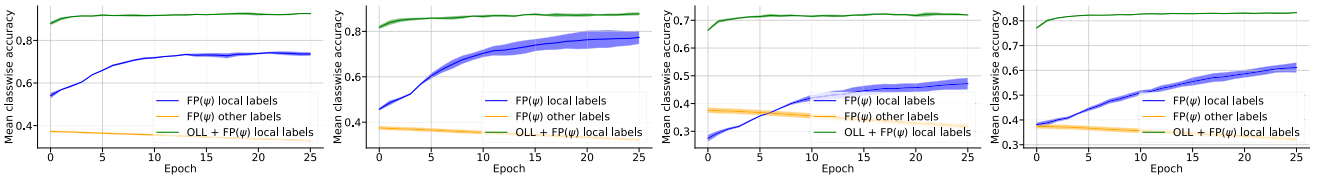

FIGURE 12. Comparison of three metrics regarding classwise accuracy on four random clients of the iNaturalist-Pers-Geo-100 dataset. The metrics compared are 1) the average classwise accuracy of the local labels for the FedAvg + FP(ψ) strategy (blue), 2) the average classwise accuracy of classes not present in the local client for the FedAvg + FP(ψ) strategy (orange), and 3) the average classwise accuracy of the local labels for the OLL(FedAvg) + FP(ψ) strategy (green). The average classwise accuracy of classes not present in the local client for the OLL(FedAvg) + FP(ψ) strategy is not included in the visualization, as it is consistently zero because of how the OLL classifier is constructed. All classwise accuracies are evaluated on the original test set of the iNaturalist dataset.

confounding factors for the classifier, such as the interference of potentially wrong-predicting classes outside the local distribution, saving computational costs, and improving final accuracy.

VI. CONCLUSION

In this work, we introduced Fed3R, a novel closed-form classifier for FL and PFL that is inherently robust to statistical heterogeneity. Fed3R leverages a pre-trained feature extractor and efficiently constructs a classifier in a non-gradient-based manner, providing a strong foundation for subsequent personalization. This approach not only accelerates training and reduces communication costs but also mitigates the detrimental effects of data recency bias and client drift. Furthermore, we demonstrated the effectiveness of Fed3R as a robust initialization for further fine-tuning with various FL and PFL algorithms. By providing a stable starting point, Fed3R facilitates faster convergence and improved performance, particularly in challenging, heterogeneous environments.

To further enhance personalization, we proposed OLL, a novel pre-processing technique that simplifies the local classifier by focusing solely on locally relevant classes. OLL effectively prevents misclassifications to classes out of the local distribution, leading to improved efficiency and performance in personalized learning.

Our comprehensive empirical evaluation on real-world cross-device datasets demonstrated the significant advantages of Fed3R and OLL in accelerating training, reducing communication and computation costs, and improving performance in both FL and PFL scenarios. These contributions pave the way for more efficient and robust FL systems, particularly in challenging, heterogeneous environments.

Future work may explore extending these methods to other learning paradigms, such as Continual Learning, where mitigating catastrophic forgetting is a crucial challenge. For instance, Fed3R could be easily extended to the Federated Continual Learning setting [100], where clients have access to streams of data and the same data points accessible in a given round of training may not be available again in future rounds. Indeed, by allowing clients to send their statistics to the server multiple times, one for each new stream of data, the server can always exactly reconstruct the optimal centralized RR solution without suffering from any catastrophic forgetting. Investigating the theoretical properties of Fed3R and OLL in these contexts could further enhance their applicability and impact.

APPENDIX

This Appendix is organized as follows:

- Appendix A presents a variation of the Fed3R algorithm that exploits Random Features [94] to approximate a Kernel Ridge Regression classifier while preserving the properties of Fed3R.
- Appendix B describes the privacy and security properties of Fed3R.
- Appendix C shows how RR can be repurposed as a tool to assess the quality of the feature extractor.
- Appendix D presents additional implementation details.
- Appendix E shows additional PP experiments.
- Appendix F shows additional FP experiments.
- Appendix G explains how the communication costs and the average computation costs per client have been approximated in this work.
- Appendix H shows additional experiments with the Cifar100 dataset [97].
- Appendix I shows that Fed3R can be combined with other orthogonal methods and is effective even in the presence of domain shift.

APPENDIX A
FED3R WITH RANDOM FEATURES (FED3R-RF)

Fed3R allows the construction of a linear classifier based on the quality of the feature maps extracted by a pre-trained feature extractor. Therefore, its performance is highly dependent on the similarity of the pre-training task, on which the feature extractor was pre-trained initially, with the target task, *i.e.*, the FL or PFL task for which we aim to build a performative model. However, feature maps generated from the samples of the target task may not be simple to separate.

A possible idea is trying to construct a federated version of Kernel Ridge Regression (KRR), similarly to what we did for RR by developing Fed3R. KRR is a nonparametric learning algorithm that uses kernel functions to address the non-linearity of the input space implicitly [88], [101]. Therefore, it may improve the separability of the latent space. However, conversely to the covariance matrix A of linear RR, which has a space complexity of $\mathcal{O}(d^2)$, the space complexity of the kernel matrix is $\mathcal{O}(n^2)$. This is a prohibitive issue, as for large datasets, where $n \gg d$, it becomes impractical to store the kernel matrix or to compute the exact KRR solution.

To overcome this limitation, we introduce Federated Recursive Ridge Regression with Random Features (Fed3R-RF). This algorithm exploits random features KRR [94]. This data-independent subsampling technique approximates the KRR solution with an explicit mapping of the latent feature space to a high-dimensional feature space with dimensionality $D > d$. This explicit mapping allows maintaining the same structure as the Fed3R algorithm and the same properties presented in Section IV-A. The complete Fed3R-RF algorithm is shown in Algorithm 3.

Algorithm 3 - Fed3R-RF

Require:
Server $\mathcal{S}$, clients $\mathcal{K}$
Fixed pre-trained feature extractor $\varphi : \mathcal{X} \to \mathbb{R}^d$
Random features $\omega \in \mathbb{R}^{d \times D}$
Random features mapping function $\Omega(\cdot; \omega)$ as in [94]
$\mathcal{S}$ initializes $A_0 = \lambda I_D$ and $b_0 = 0_{D \times C}$
for each client $k \in \mathcal{K}$ in parallel **do**
  $\hat{Z}_k = \Omega(\varphi(X_k); \omega), \quad A_k = \hat{Z}_k^\top \hat{Z}_k$
  $b_k^c = \sum_{(x,y=c) \in \mathcal{D}_k} \Omega(\varphi(x); \omega), \forall c \in \mathcal{C}_k$
  Send A_k and b_k^c $\forall c \in \mathcal{C}_k$
end for
Every time $\mathcal{S}$ receives A_k and b_k:
  $A_{t+1} = A_t + A_k, \quad b_{t+1}^c = b_t^c + b_k^c \ \ \forall c \in \mathcal{C}_k$
  $W_{t+1} = A_{t+1}^{-1} b_t$
  Normalize W: $W_{t+1}^c \leftarrow W_{t+1}^c / \left\| W_{t+1}^c \right\| \ \ \forall c \in \mathcal{C}$

For the kernel, we choose a random features approximation of an RBF kernel $k(z, \zeta) = e^{-\frac{\|z-\zeta\|^2}{2\sigma^2}}$, $z, \zeta \in \mathbb{R}^d$. We found that the best value for the best value for the hyper-parameter σ in the centralized Google Landmarks setting is $\sigma = 1000$. We decided to use this value for all the Fed3R-RF experiments.

APPENDIX B
ON THE PRIVACY PROPERTIES OF FED3R

Fed3R builds a classifier leveraging local client data. Rather than computing gradient updates, Fed3R computes local statistics (the local covariance matrix A_k and vectors b_k^c). It does so locally and communicates only aggregated statistics, never transmitting any raw client data to the server. Therefore, Fed3R fully adheres to the standard FL setting [1], [8], where the server does not have access to the local private data of the clients, thus reducing privacy risks and security risks compared with distributed learning solutions in which the client's data can instead be freely accessed.

Furthermore, similarly to Fed3R, prototype-based FL methods also transfer aggregated statistics (the prototypes) instead of model updates between the server and the clients. As noted in [45] and [102], this strategy results in improved privacy compared to other FL algorithms. In particular, [102] (FedProto) highlights that communicating prototypes naturally improves privacy since they are computed by averaging the low-dimensional latent representations of the samples from the same class, which is an irreversible process. Thus, attackers cannot reconstruct raw data from prototypes. Likewise, in Fed3R, clients only share the $(A_k; b_k^c \ \forall c \in \mathcal{C}_k)$ statistics with the server, which are aggregated values. Specifically, A_k is the sum of the outer products of all the latent feature vectors computed on the local dataset, while each b_k^c vector is the sum of all the latent feature vectors belonging to class c (that is, b_k^c is equivalent to the

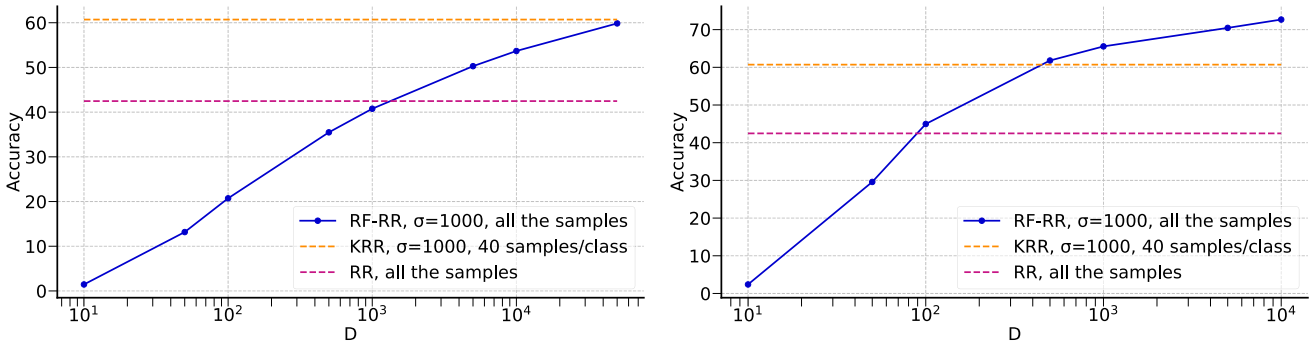

FIGURE 13. Evaluation of ridge regression with random features (RF−RR in the plots) with different values of *D*, kernel ridge regression (KRR in the plots) and linear ridge regression (RR in the plots), in the centralized Google Landmarks scenario. For KRR, we use only at most 40 samples per class because of computational limitations. The results we show here are without the final normalization step of Fed3R−RF (and Fed3R). On the left, we show the results using the feature extractor pre-trained on ImageNet. On the right, we show the results on the feature extractor fine-tuned on Google Landmarks. In both cases, increasing *D* allows for achieving higher performances, as KRR is better approximated. However, this comes at the cost of more computation and communication for Fed3R−RF.

non-averaged prototype of client k for class c):

$$A_k = \sum_{(x,y)\in\mathcal{D}_k} \varphi(x)\varphi(x)^\top, \quad b_k^c = \sum_{(x,y=c)\in\mathcal{D}_k} \varphi(x)\ \forall c \in \mathcal{C}_k.$$

Therefore, Fed3R yields the same privacy benefits as FedProto [102], and its risks in terms of privacy are at worst comparable with those of any other FL algorithm whose focus is not centered on privacy guarantees (e.g., [1], [5-7]), but on improving other key properties for the FL scenario, such as accuracy, robustness to heterogeneity, and efficiency. As outlined in Section I-A, our manuscript's key contribution is the introduction of two novel methods, Fed3R and OLL, which prove to be robust, cost-effective, and accurate in real-world cross-device FL and Personalized FL settings.

Besides, note that a distinct and substantial body of research focuses on the security and privacy properties of FL systems [103], [104], including privacy-enhancing techniques and countermeasures to malicious attacks. Indeed, the privacy of any FL algorithm, including Fed3R, can be directly improved by combining it with methods such as the Secure Aggregation protocol [105], Differential Privacy [106], [107], homomorphic encryption [108], or secure multi-party computation [109], among others.

APPENDIX C
FEATURE QUALITY EVALUATION THROUGH FED3R

Thanks to its closed-form formulation, RR or Fed3R classifiers can also be used as tools to easily assess the quality of feature extractors and the linear separability of the latent space. Indeed, it is possible to execute the Fed3R algorithm at the end of the training to evaluate the quality of the feature extractor. In fact, the quality of these linear classifiers depends only on the separability of the latent feature space. Therefore, high RR or Fed3R accuracies are proportional to the quality of the feature maps.

Table 7 compares the feature extractor parameters of different FL algorithms at convergence using Fed3R. The results indicate an improvement in the quality of the features learned when the classifier is initialized with Fed3R parameters. Indeed, Fed3R helps stabilize the training process, mitigating destructive interference and forgetting resulting from heterogeneity. Specifically, Fed3R+FT($\mathcal{M}$) and Fed3R+FT(φ) consistently achieve higher Fed3R accuracy (here, we refer to the accuracy of a new Fed3R classifier initialized using the final, trained parameters of the feature extractor) than the corresponding baseline with random classifier initialization on both the Google Landmarks and iNaturalist datasets and for all the FT algorithms. Moreover, Fed3R+FT(φ) always outperforms Fed3R+FT($\mathcal{M}$), as fixing the classifier eliminates the classifier bias issue entirely.

Moreover, Table 7 also shows that Fed3R or Fed3R−RF can provide the highest accuracy even if applied on feature extractors after the training, envisioning possible future directions for the applicability of the Fed3R and Fed3R−RF algorithms. Indeed, the Fed3R−RF−10k classifier, constructed using the features that can be extracted from trained feature extractors with different combinations of strategies for Phase 1 and 2, consistently provides better results than the softmax classifier.

APPENDIX D
ADDITIONAL IMPLEMENTATION DETAILS

All our experiments have been performed with three distinct random seeds using NVIDIA A100-SXM4-40GB GPUs. We provide all the results in the form *mean* ± *std*, where *mean* and *std* are the arithmetic mean and the standard deviation of the experiments on the three different random seeds. In the figures, the confidence intervals light regions correspond to the intervals between *mean-std* and *mean+std*.

Additional details on the datasets are provided in Table 8. Experiments on the Cifar100 dataset are shown in Section H.

In all our experiments, we augment the training images with the same data augmentation strategies adopted to pre-train our model. In particular, we apply a random resized crop to the training images, to a dimension of 224 × 224 for

TABLE 7. Quality of the feature extractors measured at convergence using `Fed3R` and `Fed3R-RF`. These results can also be interpreted as an application of `Fed3R` (`Fed3R-RF`) using fine-tuned parameters of the feature-extractor to enhance predictive performance further. Values represent the percentage accuracy.

Dataset	ψ init	FT alg	FT $(\cdot)$	softmax	Fed3R	Fed3R-RF-5k	Fed3R-RF-10k
iNat.	-	FedAvg	$\mathcal{M}$	39.5 ± 3.2	53.1 ± 0.9	55.3 ± 1.1	56.7 ± 1.0
	Fed3R	FedAvg	$\mathcal{M}$	49.0 ± 0.6	52.2 ± 0.3	53.7 ± 0.2	54.9 ± 0.3
	Fed3R	FedAvg	φ	50.8 ± 0.2	54.6 ± 0.1	55.5 ± 0.1	**56.8 ± 0.2**
Google Landmarks	-	FedAvg	$\mathcal{M}$	57.7 ± 1.2	58.8 ± 2.1	64.4 ± 0.2	66.5 ± 0.2
	Fed3R	FedAvg	$\mathcal{M}$	64.1 ± 0.4	59.6 ± 0.2	62.8 ± 0.1	**64.6 ± 0.1**
	Fed3R	FedAvg	φ	59.6 ± 0.2	62.1 ± 0.1	62.5 ± 0.1	**64.6 ± 0.1**
	-	Scaffold	$\mathcal{M}$	63.4 ± 0.9	57.0 ± 1.9	66.5 ± 0.5	68.6 ± 0.5
	Fed3R	Scaffold	$\mathcal{M}$	67.4 ± 0.3	61.8 ± 0.1	64.4 ± 0.2	66.3 ± 0.1
	Fed3R	Scaffold	φ	63.4 ± 0.1	64.3 ± 0.2	66.7 ± 0.1	**68.7 ± 0.2**

TABLE 8. Additional information about the employed datasets.

Dataset Type	Dataset	Split	Avg. samples per client	K	C	Tot. number of samples
FL	iNaturalist	Users-120K	13.0	9275	1203	120300
		Geo-100	33.4	3606	1203	120300
		Geo-300	99.6	1208	1203	120300
		Geo-1K	326.9	368	1203	120300
		Geo-3K	889.7	135	1203	120300
	Google Landmarks	Users-160K	119.9	1262	2028	151373
	Cifar100	$\alpha = 0.0$	500	100	100	50000
		$\alpha = 0.1$	500	100	100	50000
		$\alpha = 0.5$	500	100	100	50000
		$\alpha = 1.0$	500	100	100	50000
		$\alpha = 10.0$	500	100	100	50000
		$\alpha = 1000.0$	500	100	100	50000
PFL	Google Landmarks	Pers-Users-160K	80.2	823	2028	83255
	iNaturalist	Pers-Geo-100	22.7	2714	1203	66165

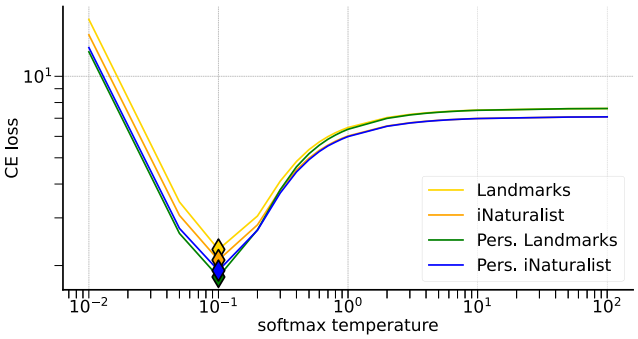

FIGURE 14. Tuning of the best temperature value for the softmax classifier, after initializing its parameters with the `Fed3R` parameters, on 4 different Google Landmarks and iNaturalist scenarios. The y-axis shows the average CE loss on the training set. The temperature value that guarantees the lowest CE loss has been chosen for all our experiments. The best value is consistently 0.1.

all the experiments except the Cifar100 experiments, where we random resize crop to 32×32.

Moreover, in all our experiments, we use SGD as the local client optimizer, with a weight decay of 4×10^{-5}. The local learning rate is 0.1 for the experiments without `Fed3R` initialization and `OLL` pre-processing. In case the classifier is initialized with `Fed3R` or there is `OLL` pre-processing before `FP`, the original learning rate is multiplied by 0.1 for each. We choose a batch size of 50 samples for the FL experiments and 64 for the PFL experiments. In all the scenarios, we choose 1 as the number of local epochs, except for the iNaturalist-User-120K, for which we choose 5.

For all our FL and PFL experiments scheduled with rounds, we sample 10 clients per round. For our `Fed3R` experiments, we choose $\lambda = 0.01$, as this value provided the best centralized RR results, although other values provided only slightly lower performance. Figure 14 shows the temperature tuning across four different Google Landmarks and iNaturalist scenarios for the initialization of the classifier parameters with the `Fed3R` ones. We selected 0.1, as this value consistently provided the lowest average training CE loss.

Figure 15 compares the best results of FL algorithms in the Google Landmarks and iNaturalist FL scenarios. We tuned all the algorithms using all the possible combinations of server learning rate $\in \{1.0, 0.1\}$, server momentum $\in \{0.9, 0.0\}$, client learning rate $\in \{0.1, 0.01\}$ and client weight decay $\in \{4 \times 10^{-5}, 0.0\}$, and we show only the best results. As it is possible to observe, `FedDyn` and `Mime` fail to converge, while `Scaffold` is extremely slow on iNaturalist. Regarding the PFL experiments, we find that the best hyper-parameters for `Ditto`, `FedSim`, `FedAlt`, and `pFedMe` are the same as `FedAvg`.

TABLE 9. Accuracy (%) of our methods compared with `FedNCM` [52].

Dataset	Fed3R	Fed3R-RF-5k	Fed3R-RF-10k	FedNCM
iNaturalist	45.1 ± 0.0	46.8 ± 0.0	**47.6 ± 0.0**	32.2 ± 0.0
Google Landmarks	49.6 ± 0.0	53.9 ± 0.0	**56.6 ± 0.0**	36.2 ± 0.0

Finally, in Table 9, we compare our `Fed3R` and `Fed3R-RF` classifiers with another, closed-form classifier from [52], based on Nearest Class Means, showing that our classifiers consistently provided better performance. As already shown in Table 2 in the main paper, `Fed3R` outperforms `FedNCM` by approximately 13 percentage points in accuracy on both the Google Landmarks and iNaturalist datasets. The performance gap widens further when comparing `Fed3R-RF-5k` or `Fed3R-RF-10k` with `FedNCM`, with differences reaching up to 20 percentage points for Google Landmarks and about 15 percentage points for iNaturalist.

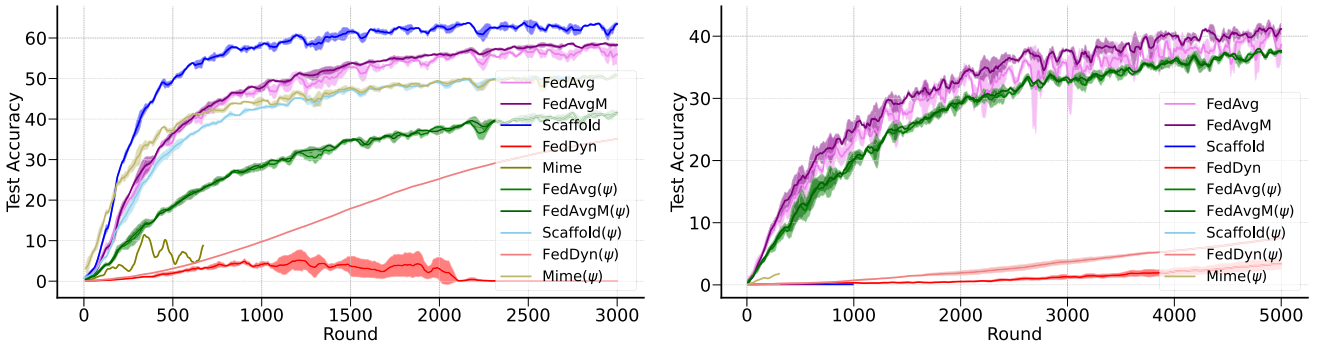

FIGURE 15. Best results of several FL baselines. On the left, it is possible to see the Google Landmarks results, while the iNaturalist results are shown on the right.

APPENDIX E
PARTIAL PERSONALIZATION COMPLETE RESULTS

In this Section, we present the complete results for the PP phase presented in Section V-D. Tables 10 and 11 provide all the final accuracies for the PP experiments. Moreover, in Tables 12 and 13, we show the communication and computation costs to achieve 70% accuracy for the same experiments in the Google Landmarks setting, while in Tables 14 and 15, we show the communication and computation costs to achieve 70% accuracy for the same experiments in the iNaturalist setting. As it is possible to see, Fed3R+PP experiments are the cheapest in terms of costs, as the expensive FT phase is not performed.

APPENDIX E
FULL PERSONALIZATION COMPLETE RESULTS

In this Section, we present the complete results for the FP phase presented in Section V-D. Tables 10 and 11 provide all the results for FP without PP phase. Moreover, in Tables 12 and 13 we show the results for the full pipeline, with all the 4 phases. The best result is achieved by the Fed3R+FedAvg+FedSim(ψ)+FP(φ) experiment, with a final weighted accuracy of 88.1%. As already seen in other contexts, the best results are achieved with Fed3R initialization. Interestingly, the best strategy is to have the classifier as local parameters first during the PP phase and then locally fine-tune the feature extractor only during the FP phase while keeping the classifier fixed.

APPENDIX G
COMMUNICATION AND COMPUTATION COSTS

In this Section, we describe how we compute the communication and computation costs for our experiments.

A. TOTAL COMMUNICATION COSTS

In our plots, such as the ones of Figure 9 or Figure 10, we describe the communication in terms of total communication required to achieve any given target accuracy. To find this value, we begin by estimating the communication costs for each client in each round. The total communication cost for each round consists of two components: downstream costs, which involve communication from the server to the clients, and upstream costs, which involve communication from the clients to the server. These costs can vary depending on the chosen method.

After collecting this information, we calculate the total costs per round by multiplying the costs per round for each client by the number of clients sampled in that round, for the methods that are scheduled in rounds. Instead, for our Fed3R and Fed3R-RF algorithms, we evaluate the cumulative costs for every client that has already shared its statistics with the server. In all communication cost plots, we multiply the final values by 4 to express the total cost in bytes, as we assume that all parameters are stored as FP32 values, which are 4 bytes each.

Let $\bar{C} = \frac{1}{K}\sum_{k\in\mathcal{K}} C_k$ be the average number of classes accessible to each client. Additionally, let c, b, and m represent the sizes of the classifier, the feature extractor, and the whole model, respectively. Since the classifier is a linear layer, its size is equal to the product of the latent feature dimensionality and the number of classes in the dataset; thus, $c = dC$. Therefore, the total model size can be expressed as $m = b + dC$. As already specified in Section III-C, the communication required for any FP method is 0. Table 22 summarizes how the downstream and upstream costs are calculated for each algorithm. In particular, for the non-gradient-based methods, *i.e.,* Fed3R, Fed3R-RF and FedNCM, the upstream communication costs also represent the additional required storage for each client. This storage is irrisory for all the three algorithms. Indeed, for instance, training on iNaturalist requires each client to have an additional storage of 0.85 MB, an irrisory quantity for modern architectures, even considering low-budget devices.

B. AVERAGE COMPUTATION PER CLIENT

To estimate the average computation per client, let $\boldsymbol{x}$ denote the number of times a specific client is sampled and $\mathcal{T}$ denote the total average cost per round for a single client. The cumulative average cost $\mathcal{T}_t$ from round 1 to round t is proportional to the expected number of times a specific client

TABLE 10. Google Landmarks PP final weighted accuracies.

ψ init	FT alg	FT acc	FedSim (ψ)	FedAlt (ψ)	pFedMe (ψ)	FedSim (φ)	FedAlt (φ)	pFedMe (φ)
✗	FedAvg	61.2 ± 1.0	79.4 ± 0.0	79.5 ± 0.0	66.7 ± 0.0	79.7 ± 0.1	79.7 ± 0.0	67.0 ± 0.2
✗	FedAvg-1k	47.6 ± 1.0	66.5 ± 0.2	66.6 ± 0.2	57.2 ± 0.1	74.1 ± 0.0	74.2 ± 0.0	56.1 ± 0.2
Fed3R	✗	49.9 ± 0.0	72.9 ± 0.1	72.2 ± 0.3	55.0 ± 0.2	74.1 ± 0.1	73.2 ± 0.2	53.9 ± 0.4
Fed3R	FedAvg	68.1 ± 0.4	79.9 ± 0.1	79.4 ± 0.1	69.7 ± 0.4	**80.4 ± 0.0**	80.3 ± 0.1	69.7 ± 0.0
Fed3R	FedAvg-1k	63.0 ± 0.3	77.0 ± 0.3	76.2 ± 1.0	45.5 ± 0.0	78.6 ± 0.1	78.4 ± 0.1	41.2 ± 0.3
Fed3R	FedAvg (φ)	68.2 ± 0.1	79.9 ± 0.1	79.5 ± 0.1	70.0 ± 0.0	**80.4 ± 0.0**	**80.4 ± 0.0**	69.6 ± 0.0

TABLE 11. iNaturalist PP final weighted accuracies.

ψ init	FT alg	FT acc	FedSim (ψ)	FedAlt (ψ)	pFedMe (ψ)	FedSim (φ)	FedAlt (φ)	pFedMe (φ)
✗	FedAvg	46.3 ± 0.4	70.9 ± 0.2	71.0 ± 0.1	52.3 ± 0.1	67.4 ± 0.2	67.9 ± 0.1	53.0 ± 0.0
✗	FedAvg-1k	30.0 ± 0.3	40.7 ± 0.1	40.7 ± 0.1	38.3 ± 0.1	52.1 ± 0.1	53.1 ± 0.4	37.9 ± 0.0
Fed3R	✗	58.0 ± 0.0	75.1 ± 0.0	73.8 ± 0.1	59.8 ± 0.1	72.9 ± 0.1	74.4 ± 0.1	59.0 ± 0.0
Fed3R	FedAvg	67.3 ± 0.2	**78.5 ± 0.1**	77.9 ± 0.1	68.8 ± 0.0	76.9 ± 0.1	78.0 ± 0.0	68.3 ± 0.0
Fed3R	FedAvg-1k	63.5 ± 0.1	77.3 ± 0.1	76.3 ± 0.1	42.1 ± 0.1	75.5 ± 0.1	76.7 ± 0.1	41.7 ± 0.1
Fed3R	FedAvg (φ)	63.1 ± 0.1	77.5 ± 0.0	76.6 ± 0.0	64.5 ± 0.0	77.2 ± 0.1	76.9 ± 0.1	64.6 ± 0.0

TABLE 12. Google Landmarks PP total communication costs to achieve 70% weighted accuracy. N/A indicates that the experiment never reaches the target accuracy.

ψ init	FT alg	FT costs	FedSim (ψ)	FedAlt (ψ)	pFedMe (ψ)	FedSim (φ)	FedAlt (φ)	pFedMe (φ)
✗	FedAvg	N/A	1.2TB ± 720.7MB	1.2TB ± 720.7MB	N/A	1.2TB ± 98.1MB	1.2TB ± 169.8MB	N/A
✗	FedAvg-1k	N/A	413.3GB ± 1.4GB	411.7GB ± 1.3GB	N/A	448.2GB ± 2.4GB	426.6GB ± 4.9GB	N/A
Fed3R	✗	N/A	**39.6GB ± 3.2GB**	48.5GB ± 2.6GB	N/A	100.3GB ± 2.7GB	77.4GB ± 1.2GB	N/A
Fed3R	FedAvg	N/A	1.1TB ± 851.4MB	1.1TB ± 819.9MB	N/A	1.1TB ± 196.1MB	1.1TB ± 196.1MB	N/A
Fed3R	FedAvg-1k	N/A	392.2GB ± 280.9MB	392.6GB ± 280.9MB	N/A	407.0GB ± 1.1GB	397.1GB ± 588.3MB	N/A
Fed3R	FedAvg (φ)	N/A	519.8GB ± 265.8MB	519.9GB ± 283.8MB	669.3GB ± 4.4GB	523.6GB ± 107.5MB	521.8GB ± 107.5MB	N/A

TABLE 13. Google Landmarks PP average computation per client to achieve 70% weighted accuracy. N/A indicates that the experiment never reaches the target accuracy.

ψ init	FT alg	FT costs	FedSim (ψ)	FedAlt (ψ)	pFedMe (ψ)	FedSim (φ)	FedAlt (φ)	pFedMe (φ)
✗	FedAvg	N/A	19.2 TFLOPs ± 25.9 GFLOPs	19.1 TFLOPs ± 25.9 GFLOPs	N/A	18.7 TFLOPs ± 985.6 MFLOPs	18.7 TFLOPs ± 1.7 GFLOPs	N/A
✗	FedAvg-1k	N/A	7.4 TFLOPs ± 50.5 GFLOPs	7.3 TFLOPs ± 46.9 GFLOPs	N/A	6.9 TFLOPs ± 23.9 GFLOPs	6.6 TFLOPs ± 49.0 GFLOPs	N/A
Fed3R	✗	N/A	1.4 TFLOPs ± 113.6 GFLOPs	1.7 TFLOPs ± 93.7 GFLOPs	N/A	**1.0 TFLOPs ± 27.6 GFLOPs**	782.8 GFLOPs ± 12.0 GFLOPs	N/A
Fed3R	FedAvg	N/A	18.7 TFLOPs ± 30.6 GFLOPs	18.7 TFLOPs ± 29.4 GFLOPs	N/A	18.6 TFLOPs ± 2.0 GFLOPs	18.6 TFLOPs ± 2.0 GFLOPs	N/A
Fed3R	FedAvg-1k	N/A	6.6 TFLOPs ± 10.1 GFLOPs	6.6 TFLOPs ± 10.1 GFLOPs	N/A	6.4 TFLOPs ± 10.7 GFLOPs	6.3 TFLOPs ± 5.9 GFLOPs	N/A
Fed3R	FedAvg (φ)	N/A	18.7 TFLOPs ± 9.5 GFLOPs	18.7 TFLOPs ± 10.2 GFLOPs	29.5 TFLOPs ± 317.8 GFLOPs	18.6 TFLOPs ± 1.7 GFLOPs	18.6 TFLOPs ± 1.7 GFLOPs	N/A

TABLE 14. iNaturalist PP total communication costs to achieve 70% weighted accuracy. N/A indicates that the experiment never reaches the target accuracy.

ψ init	FT alg	FT costs	FedSim (ψ)	FedAlt (ψ)	pFedMe (ψ)	FedSim (φ)	FedAlt (φ)	pFedMe (φ)
✗	FedAvg	N/A	1.0TB ± 3.8GB	1.0TB ± 3.5GB	N/A	N/A	N/A	N/A
✗	FedAvg-1k	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Fed3R	✗	N/A	54.0GB ± 996.3MB	66.5GB ± 81.1MB	N/A	78.0GB ± 1.7GB	**53.0GB ± 303.7MB**	N/A
Fed3R	FedAvg	N/A	900.4GB ± 720.9MB	902.6GB ± 401.9MB	N/A	904.1GB ± 623.3MB	898.9GB ± 406.6MB	N/A
Fed3R	FedAvg-1k	N/A	324.6GB ± 292.3MB	329.8GB ± 405.4MB	N/A	336.3GB ± 2.0GB	323.3GB ± 773.3MB	N/A
Fed3R	FedAvg (φ)	N/A	545.6GB ± 243.2MB	548.9GB ± 1.5GB	N/A	542.2GB ± 861.1MB	540.2GB ± 998.9MB	N/A

TABLE 15. iNaturalist PP average computation per client to achieve 70% weighted accuracy. N/A indicates that the experiment never reaches the target accuracy.

ψ init	FT alg	FT costs	FedSim (ψ)	FedAlt (ψ)	pFedMe (ψ)	FedSim (φ)	FedAlt (φ)	pFedMe (φ)
✗	FedAvg	N/A	1.7 TFLOPs ± 9.7 GFLOPs	1.7 TFLOPs ± 8.9 GFLOPs	N/A	N/A	N/A	N/A
✗	FedAvg-1k	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Fed3R	✗	N/A	137.7 GFLOPs ± 2.6 GFLOPs	170.0 GFLOPs ± 209.2 MFLOPs	N/A	93.6 GFLOPs ± 2.1 GFLOPs	**63.5 GFLOPs ± 365.6 MFLOPs**	N/A
Fed3R	FedAvg	N/A	1.4 TFLOPs ± 1.7 GFLOPs	1.4 TFLOPs ± 911.7 MFLOPs	N/A	1.4 TFLOPs ± 760.5 MFLOPs	1.3 TFLOPs ± 487.5 MFLOPs	N/A
Fed3R	FedAvg-1k	N/A	517.5 GFLOPs ± 754.1 MFLOPs	530.8 GFLOPs ± 1.0 GFLOPs	N/A	492.2 GFLOPs ± 2.4 GFLOPs	476.6 GFLOPs ± 930.8 MFLOPs	N/A
Fed3R	FedAvg (φ)	N/A	1.4 TFLOPs ± 627.5 MFLOPs	1.4 TFLOPs ± 4.0 GFLOPs	N/A	1.4 TFLOPs ± 1.0 GFLOPs	1.4 TFLOPs ± 1.2 GFLOPs	N/A

TABLE 16. Google Landmarks FP results without PP phase.

ψ init	FT alg	FT acc	FP (ψ)		FP (φ)		FP ($\mathcal{M}$)	
			SGD	Ditto	SGD	Ditto	SGD	Ditto
✗	FedAvg	61.2 ± 1.0	78.3 ± 0.1	78.8 ± 0.0	78.3 ± 0.2	78.4 ± 0.1	80.0 ± 0.0	80.0 ± 0.0
✗	FedAvg-1k	47.6 ± 1.0	72.6 ± 0.1	73.0 ± 0.2	72.5 ± 0.1	71.3 ± 0.3	71.7 ± 0.3	71.7 ± 0.0
Fed3R	✗	49.9 ± 0.0	68.4 ± 0.2	68.5 ± 0.1	68.9 ± 0.1	68.7 ± 0.1	69.7 ± 0.1	69.6 ± 0.2
Fed3R	FedAvg	68.1 ± 0.4	80.0 ± 0.0	79.4 ± 0.1	79.5 ± 0.1	79.4 ± 0.1	80.0 ± 0.1	**80.2 ± 0.1**
Fed3R	FedAvg-1k	63.0 ± 0.3	77.6 ± 0.2	77.7 ± 0.1	76.9 ± 0.1	76.8 ± 0.1	77.4 ± 0.2	77.6 ± 0.2
Fed3R	FedAvg (φ)	68.2 ± 0.1	80.1 ± 0.1	**80.2 ± 0.0**	79.5 ± 0.0	79.4 ± 0.0	80.0 ± 0.1	**80.2 ± 0.1**

TABLE 17. iNaturalist FP results without PP phase.

ψ init	FT alg	FT acc	FP (ψ)		FP (φ)		FP ($\mathcal{M}$)	
			SGD	Ditto	SGD	Ditto	SGD	Ditto
✗	FedAvg	46.3 ± 0.4	75.2 ± 0.1	76.7 ± 0.1	75.5 ± 0.2	76.8 ± 0.1	78.1 ± 0.1	83.5 ± 0.1
✗	FedAvg-1k	30.0 ± 0.3	68.0 ± 0.0	69.1 ± 0.1	67.1 ± 0.0	66.8 ± 0.0	66.8 ± 0.0	68.1 ± 0.1
Fed3R	✗	58.0 ± 0.0	77.4 ± 0.1	77.5 ± 0.1	78.6 ± 0.2	78.6 ± 0.2	77.1 ± 0.1	77.3 ± 0.1
Fed3R	FedAvg	67.3 ± 0.2	83.8 ± 0.0	84.0 ± 0.1	84.1 ± 0.2	**84.2 ± 0.1**	83.3 ± 0.0	83.5 ± 0.1
Fed3R	FedAvg-1k	63.5 ± 0.1	81.8 ± 0.1	82.0 ± 0.1	82.3 ± 0.1	82.3 ± 0.2	81.4 ± 0.1	81.5 ± 0.1
Fed3R	FedAvg (φ)	63.1 ± 0.1	81.6 ± 0.1	81.7 ± 0.0	80.5 ± 0.0	80.6 ± 0.0	79.5 ± 0.1	79.6 ± 0.0

TABLE 18. Google Landmarks FP results with PP (ψ) phase.

ψ init	FT alg	FedSim (ψ)				FedAlt (ψ)			
		✗	FP (ψ)	FP (φ)	FP ($\mathcal{M}$)	✗	FP (ψ)	FP (φ)	FP ($\mathcal{M}$)
✗	FedAvg	79.4 ± 0.0	79.7 ± 0.0	**80.4 ± 0.0**	**80.4 ± 0.0**	79.5 ± 0.0	79.7 ± 0.0	80.2 ± 0.0	80.2 ± 0.0
✗	FedAvg-1k	72.1 ± 0.1	72.9 ± 0.0	72.9 ± 0.0	73.0 ± 0.1	72.4 ± 0.1	72.7 ± 0.1	72.7 ± 0.2	72.7 ± 0.2
Fed3R	✗	72.9 ± 0.1	72.9 ± 0.1	73.0 ± 0.1	73.5 ± 0.1	72.2 ± 0.3	72.4 ± 0.2	72.9 ± 0.0	72.3 ± 0.1
Fed3R	FedAvg	79.9 ± 0.1	79.8 ± 0.1	80.1 ± 0.0	79.5 ± 0.2	79.4 ± 0.1	79.5 ± 0.1	79.8 ± 0.2	79.2 ± 0.0
Fed3R	FedAvg-1k	77.0 ± 0.3	77.0 ± 0.2	77.5 ± 0.1	76.9 ± 0.2	76.2 ± 1.0	76.4 ± 0.6	77.2 ± 0.2	76.5 ± 0.1
Fed3R	FedAvg (φ)	79.9 ± 0.1	79.8 ± 0.1	80.1 ± 0.0	79.5 ± 0.2	79.5 ± 0.1	79.5 ± 0.1	79.8 ± 0.2	79.2 ± 0.0

TABLE 19. iNaturalist FP results with PP (ψ) phase.

ψ init	FT alg	FedSim (ψ)				FedAlt (ψ)			
		✗	FP (ψ)	FP (φ)	FP ($\mathcal{M}$)	✗	FP (ψ)	FP (φ)	FP ($\mathcal{M}$)
✗	FedAvg	70.9 ± 0.2	82.5 ± 0.2	81.8 ± 0.2	80.7 ± 3.7	71.0 ± 0.1	82.4 ± 0.0	81.7 ± 0.3	83.3 ± 0.1
✗	FedAvg-1k	66.2 ± 0.1	79.8 ± 0.2	73.0 ± 0.1	80.9 ± 0.2	65.6 ± 0.1	79.8 ± 0.0	72.7 ± 0.1	80.9 ± 0.2
Fed3R	✗	75.1 ± 0.0	85.6 ± 0.2	86.4 ± 0.1	85.3 ± 0.1	73.8 ± 0.1	85.0 ± 0.1	85.7 ± 0.2	84.6 ± 0.1
Fed3R	FedAvg	78.5 ± 0.1	87.7 ± 0.1	**88.1 ± 0.1**	87.4 ± 0.1	77.9 ± 0.1	87.3 ± 0.1	88.0 ± 0.1	87.0 ± 0.1
Fed3R	FedAvg-1k	77.3 ± 0.1	86.9 ± 0.1	87.4 ± 0.1	86.9 ± 0.1	76.3 ± 0.1	86.4 ± 0.1	87.1 ± 0.1	86.1 ± 0.1
Fed3R	FedAvg (φ)	77.5 ± 0.0	87.0 ± 0.2	87.5 ± 0.1	86.4 ± 0.2	76.6 ± 0.0	86.6 ± 0.1	87.1 ± 0.1	85.8 ± 0.1

is sampled over t rounds, expressed as $\mathbb{E}[\boldsymbol{x}] = t\frac{K}{K}$. Therefore, we have $\mathcal{T}_t = \mathcal{T}\mathbb{E}[\boldsymbol{x}] = \mathcal{T}t\frac{K}{K}$.

Let F_* and B_* represent the costs of one forward pass and one backward pass of a single image through $*$, respectively. $*$ is one among $\mathcal{M}$, φ, ψ. We approximate $B_* \simeq 2F_*$ as in [52], and measure the costs in FLOPs. In addition, let $\hat{F}_\psi$ be the forward cost though the OLL classifier, which has a dimension $d \times \hat{C}$, with $\hat{C} \ll C$. Therefore, $\hat{F}_\psi \ll F_\psi$. Finally, let E denote the number of local epochs. Our estimation of the costs $\mathcal{T}$ per each method is summarized in Table 23.

APPENDIX H
CIFAR100 FL EXPERIMENTS

In addition to Google Landmarks and iNaturalist, we also run experiments on the simpler, less heterogeneous Cifar100 dataset [97] on the Dirichlet splits proposed by [35]. The

TABLE 20. Google Landmarks FP results with PP(φ) phase.

ψ init	FT alg	FedSim(φ)				FedAlt(φ)			
		✗	FP(ψ)	FP(φ)	FP($\mathcal{M}$)	✗	FP(ψ)	FP(φ)	FP($\mathcal{M}$)
✗	FedAvg	79.7 ± 0.1	80.2 ± 0.0	80.0 ± 0.0	80.3 ± 0.1	79.7 ± 0.0	80.2 ± 0.0	80.0 ± 0.0	80.3 ± 0.0
✗	FedAvg-1k	74.5 ± 0.1	75.3 ± 0.0	75.6 ± 0.1	75.6 ± 0.1	74.0 ± 0.2	74.5 ± 0.1	74.9 ± 0.1	74.8 ± 0.1
Fed3R	✗	74.1 ± 0.1	73.1 ± 0.2	74.5 ± 0.1	73.0 ± 0.1	73.2 ± 0.2	71.6 ± 0.1	73.7 ± 0.1	72.2 ± 0.1
Fed3R	FedAvg	80.4 ± 0.0	80.3 ± 0.1	**80.7 ± 0.1**	80.2 ± 0.1	80.3 ± 0.1	79.8 ± 0.1	80.5 ± 0.1	79.8 ± 0.1
Fed3R	FedAvg-1k	78.6 ± 0.1	78.1 ± 0.0	78.9 ± 0.1	78.0 ± 0.1	78.4 ± 0.1	77.4 ± 0.0	78.6 ± 0.1	77.5 ± 0.1
Fed3R	FedAvg(φ)	80.4 ± 0.0	80.3 ± 0.1	80.6 ± 0.0	80.1 ± 0.1	80.4 ± 0.0	79.9 ± 0.1	80.6 ± 0.1	79.8 ± 0.1

TABLE 21. iNaturalist FP results with PP(φ) phase.

ψ init	FT alg	FedSim(φ)				FedAlt(φ)			
		✗	FP(ψ)	FP(φ)	FP($\mathcal{M}$)	✗	FP(ψ)	FP(φ)	FP($\mathcal{M}$)
✗	FedAvg	67.4 ± 0.2	82.1 ± 0.1	79.3 ± 0.1	82.4 ± 0.1	67.9 ± 0.1	82.1 ± 0.1	79.4 ± 0.1	82.5 ± 0.2
✗	FedAvg-1k	63.6 ± 0.2	79.2 ± 0.1	76.4 ± 0.2	80.2 ± 0.1	66.7 ± 0.3	79.5 ± 0.1	78.1 ± 0.1	81.1 ± 0.0
Fed3R	✗	72.9 ± 0.1	83.4 ± 0.1	84.8 ± 0.1	82.8 ± 0.1	74.4 ± 0.1	83.5 ± 0.1	85.2 ± 0.2	83.1 ± 0.2
Fed3R	FedAvg	76.9 ± 0.1	86.8 ± 0.1	87.4 ± 0.1	86.4 ± 0.1	78.0 ± 0.0	87.1 ± 0.1	**87.8 ± 0.1**	86.7 ± 0.1
Fed3R	FedAvg-1k	75.5 ± 0.1	85.8 ± 0.0	86.5 ± 0.1	85.3 ± 0.1	76.7 ± 0.1	86.0 ± 0.1	86.9 ± 0.1	85.7 ± 0.1
Fed3R	FedAvg(φ)	77.2 ± 0.1	86.5 ± 0.0	86.1 ± 0.0	84.9 ± 0.1	76.9 ± 0.1	86.1 ± 0.1	86.1 ± 0.0	84.6 ± 0.1

TABLE 22. Estimation of communication costs of all the methods included in our experiments per each sampled client in one round. For Fed3R, Fed3R-RF and FedNCM, here we simply indicate the communication required for each client. All FP do not necessitate any communication.

Method	Downstream	Upstream	Total
Fed3R	0	$\frac{1}{2}d(d+1)+d\bar{C}$	$\frac{1}{2}d(d+1)+d\bar{C}$
Fed3R-RF	0	$\frac{1}{2}D(D+1)+D\bar{C}$	$\frac{1}{2}D(D+1)+D\bar{C}$
FedNCM	0	dC	dC
FedAvg	$b+dC$	$b+dC$	$2(b+dC)$
FedAvg(φ)	b	b	$2b$
FedAvg(ψ)	dC	dC	$2dC$
FedAvgM	$b+dC$	$b+dC$	$2(b+dC)$
FedAvgM(φ)	b	b	$2b$
FedAvgM(ψ)	dC	dC	$2dC$
Scaffold	$2(b+dC)$	$2(b+dC)$	$4(b+dC)$
Scaffold(φ)	$2b$	$2b$	$4b$
Scaffold(ψ)	$2dC$	$2dC$	$4dC$
FedSim(φ)	dC	dC	$2dC$
FedSim(ψ)	b	b	$2b$
FedAlt(φ)	dC	dC	$2dC$
FedAlt(ψ)	b	b	$2b$
pFedMe(φ)	dC	dC	$2dC$
pFedMe(ψ)	b	b	$2b$

TABLE 23. Estimation of average computation costs per client per round of all the methods included in our experiments per each sampled client in one round. For Fed3R, Fed3R-RF and FedNCM, we simply indicate the average computation costs for each client. For the FP methods, we indicate the average computation costs per client per epoch.

Method	$\mathcal{T}$
Fed3R	$F_\varphi+\frac{1}{2}d(d+1)+d\bar{C}$
Fed3R-RF	$F_\varphi+\frac{1}{2}D(D+1)+d\bar{C}$
FedNCM	F_φ
FedAvg	$3EF_{\mathcal{M}}$
FedAvg(φ)	$3EF_{\mathcal{M}}$
FedAvg(ψ)	$3E(F_\varphi+3F_\psi)$
FedAvgM	$3EF_{\mathcal{M}}$
FedAvgM(φ)	$3EF_{\mathcal{M}}$
FedAvgM(ψ)	$3E(F_\varphi+3F_\psi)$
Scaffold	$3EF_{\mathcal{M}}$
Scaffold(φ)	$3EF_{\mathcal{M}}$
Scaffold(ψ)	$3E(F_\varphi+3F_\psi)$
FedSim(φ)	$3EF_{\mathcal{M}}$
FedSim(ψ)	$3E(F_\varphi+3F_\psi)$
FedAlt(φ)	$3EF_{\mathcal{M}}$
FedAlt(ψ)	$3E(F_\varphi+3F_\psi)$
pFedMe(φ)	$6EF_{\mathcal{M}}$
pFedMe(ψ)	$6E(F_\varphi+3F_\psi)$
FP($\mathcal{M}$)	$3F_{\mathcal{M}}$
FP(φ)	$3F_{\mathcal{M}}$
FP(ψ)	$3(F_\varphi+3F_\psi)$
FP($\mathcal{M}$) + Ditto	$3F_{\mathcal{M}}$
FP(φ) + Ditto	$3F_{\mathcal{M}}$
FP(ψ) + Ditto	$3(F_\varphi+3F_\psi)$
OLL($\mathcal{M}$)	$3(3F_\varphi+3\hat{F}_\psi)$
OLL(φ)	$3(3F_\varphi+3\hat{F}_\psi)$
OLL(ψ)	$6(F_\varphi+3\hat{F}_\psi)$

level of statistical heterogeneity is controlled by a Dirichlet parameter α. A lower α is associated with a higher level of statistical heterogeneity. In the following, we will always implicitly refer to Cifar100 with $\alpha = 0$, except than when we specify a different α value. In the $\alpha = 0$ split, each client has access to only one class, and the same class is present in only that client. Details on these splits are included in Table 8. All the Cifar100 FT experiments have been run for 1.5K rounds.

Figure 16 compares Fed3R and Fed3R-RF with classical FL baselines. As it is possible to observe, both Fed3R and Fed3R-RF save communication and computation costs, although performance is not as good as the other

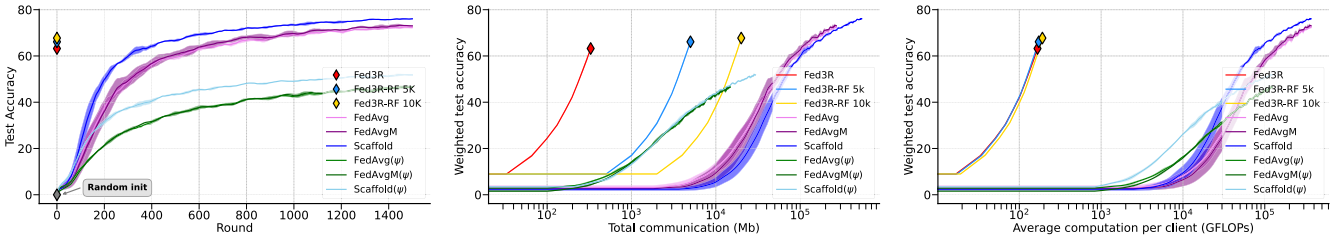

FIGURE 16. Comparison between `Fed3R`, `Fed3R-RF` and other FL baselines in the Cifar100 setting. From left to right: test accuracy by rounds, test accuracy by total communication budget, test accuracy by expected computation per client, on average.

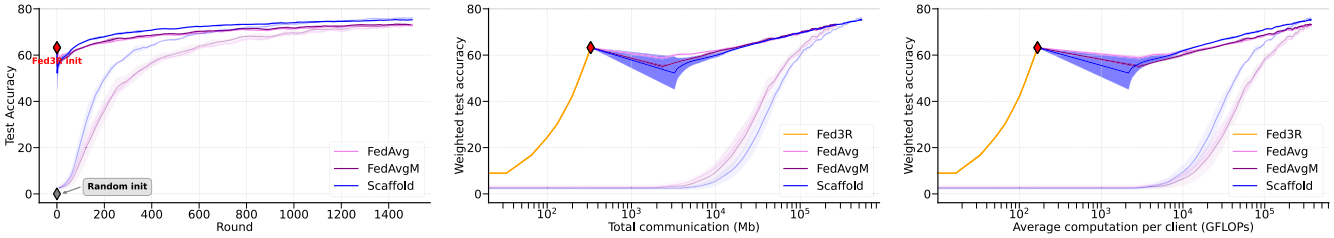

FIGURE 17. Comparison between FL algorithms on the Cifar100 dataset, and the same algorithms fine-tuned after initializing the classifier parameters using `Fed3R`. From left to right: test accuracy by rounds, test accuracy by total communication budget, test accuracy by expected computation per client, on average. The red marker indicates the `Fed3R` initialization point, which allows fine-tuning with any FL algorithm. Less intense colors are associated with the baseline algorithms without `Fed3R` initialization.

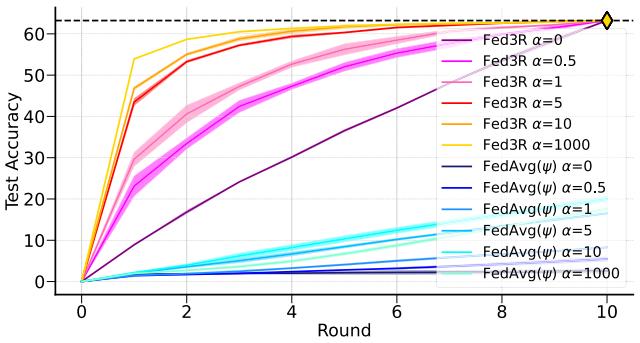

FIGURE 18. `Fed3R` vs. `FedAvg(ψ)` on different splits of Cifar100, with groups of $\kappa = 10$ clients, with different levels of statistical heterogeneity (a lower value of α indicates a higher level of statistical heterogeneity).

methods at convergence. This is because Cifar100 is a simpler dataset than Google Landmarks and iNaturalist, with few clients and low heterogeneity. Therefore, classical FL algorithms, which usually struggle in highly statistically heterogeneous settings, can reach higher accuracies here. However, Figure 17 shows how the `Fed3R` initialization and sequential fine-tuning allow for reducing costs in the early stages of the training, although the final performances are the same at convergence because of the simplicity of the scenario.

Finally, in Figure 18, we empirically show that `Fed3R` is immune to statistical heterogeneity using six distinct Cifar100 splits and that `Fed3R` allows achieving 3 times the performance achieved by `FedAvg(ψ)` in the same number of rounds `Fed3R` needs to converge, simulating rounds for `Fed3R` as in Section V-C.

TABLE 24. Accuracy (%) on the FL test set of the iNaturalist dataset, using the Pers-Geo-100 training clients, of 5 different strategies including various combinations of `Fed3R`, `FedAvg`, and `FedRDN`. The datasets are perturbed to enhance the domain shift. The severity of the perturbation is controlled by a domain shift intensity parameter δ. Best results are highlighted in **bold**, and second bests are underlined.

ψ init	FT alg	Domain shift intensity δ		
		1.2	1.5	2.0
✗	FedAvg	25.3 ± 1.2	24.9 ± 1.0	17.6 ± 2.3
✗	FedAvg + FedRDN	27.3 ± 2.6	25.7 ± 2.5	20.9 ± 0.7
Fed3R	✗	40.0 ± 0.0	33.6 ± 0.0	28.6 ± 0.0
Fed3R	FedAvg	<u>44.4 ± 1.4</u>	<u>44.2 ± 1.2</u>	<u>40.2 ± 1.6</u>
Fed3R	FedAvg + FedRDN	**47.1 ± 0.3**	**45.6 ± 2.3**	**41.7 ± 0.8**

APPENDIX I
FED3R MODULARITY AND DOMAIN SHIFT ROBUSTNESS

In this appendix section, we empirically demonstrate that `Fed3R` can be combined with other orthogonal methods to further improve performance. We have chosen `FedRDN` [110], a novel orthogonal FL method to address domain shifts in FL, as a representative pre-processing method to compare against –or combine with– `Fed3R`. Moreover, we show that `Fed3R` is robust to FL scenarios with various domain shift intensities.

A. EXPERIMENTAL SETUP

For the training, we utilize the clients from the Pers-Geo-100 split of iNaturalist. The global model is evaluated using the original iNaturalist test set.

To induce domain shift, we divide the clients into three equal-sized groups, each associated with a specific RGB channel. In the first group, we multiply the red channel of the images by a domain shift intensity factor δ, while the

green and blue channels are multiplied by the factor $1/\delta$. The domains of the other two groups are analogously shifted by enhancing the green and blue channels, respectively. We also enhance in the same manner the red, green, and blue channels of the test set over three equally sized groups.

B. RESULTS

Table 24 presents the results of our experiments on the robustness of `Fed3R` to domain shift and its compatibility with other orthogonal methods across three different levels of domain shift intensity.

We observe that `FedAvg` is significantly impacted by domain shift, resulting in declining accuracies as the intensity of the shift increases. `FedRDN` improves robustness to domain shift, mildly improving the accuracy of `FedAvg` by 1-3%. Nonetheless, `Fed3R` alone outperforms both `FedAvg` by itself and the combination of `FedAvg` with `FedRDN` by a considerable margin (e.g., +15% and +13% with respect to `FedAvg` and `FedAvg + FedRDN`, in the setting with $\delta = 1.2$). Additionally, as discussed in the main paper, the `Fed3R` classifier can be further fine-tuned with another federated learning algorithm, such as `FedAvg`; our results show that the `Fed3R + FedAvg` strategy boosts accuracy in all three scenarios. Finally, adding `FedRDN` to the `Fed3R + FedAvg` strategy leads to an additional increase in accuracy (e.g., +3% with respect to `Fed3R + FedAvg`, and +20% with respect to `FedAvg + FedRDN`, in the setting with $\delta = 1.2$).

ACKNOWLEDGMENT

This manuscript reflects only the authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them. The authors acknowledge the CINECA Award under the ISCRA Initiative for the availability of high-performance computing resources and support. They also thank the reviewers and Associate Editor for their valuable comments. This work has been mainly carried out while Marco Ciccone was with the Polytechnic University of Turin.

REFERENCES

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist.*, Jan. 2016, pp. 1273–1282.

[2] P. Kairouz et al., "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, nos. 1–2, pp. 1–210, 2021.

[3] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, "Towards personalized federated learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 12, pp. 9587–9603, Dec. 2022.

[4] T. Yang, G. Andrew, H. Eichner, H. Sun, W. Li, N. Kong, D. Ramage, and F. Beaufays, "Applied federated learning: Improving Google keyboard query suggestions," 2018, *arXiv:1812.02903*.

[5] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, Jan. 2020, pp. 3557–3568.

[6] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic controlled averaging for federated learning," in *Proc. Int. Conf. Mach. Learn.*, Oct. 2019, pp. 5132–5143.

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

[8] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. Mach. Learn. Syst.*, vol. 2, Jan. 2018, pp. 429–450.

[9] E. Ozfatura, K. Ozfatura, and D. Gündüz, "FedADC: Accelerated federated learning with drift control," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2021, pp. 467–472.

[10] Y. Liu, Y. Sun, Z. Ding, L. Shen, B. Liu, and D. Tao, "Enhance local consistency in federated learning: A multi-step inertial momentum approach," 2023, *arXiv:2302.05726*.

[11] J. Xu, S. Wang, L. Wang, and A. Chi-Chih Yao, "FedCM: Federated learning with client-level momentum," 2021, *arXiv:2106.10874*.

[12] M. G. Arivazhagan, V. Aggarwal, A. Kumar Singh, and S. Choudhary, "Federated learning with personalization layers," 2019, *arXiv:1912.00818*.

[13] V. Smith, C.-K. Chiang, M. Sanjabi, and A. Talwalkar, "Federated multi-task learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, Dec. 2017, pp. 4427–4437.

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

[15] J. Nguyen, K. Malik, M. Sanjabi, and M. Rabbat, "Where to begin? Exploring the impact of pre-training and initialization in federated learning," in *Proc. Int. Conf. Learn. Represent.*, 2023, pp. 1–17.

[16] J. Nguyen, J. Wang, K. Malik, M. Sanjabi, and M. Rabbat, "Where to begin? On the impact of pre-training and initialization in federated learning," 2022, *arXiv:2206.15387*.

[17] T. Zhou, J. Zhang, and D. H. K. Tsang, "FedFA: Federated learning with feature anchors to align features and classifiers for heterogeneous data," *IEEE Trans. Mobile Comput.*, vol. 23, no. 6, pp. 6731–6742, Jun. 2024.

[18] Z. Li, X. Shang, R. He, T. Lin, and C. Wu, "No fear of classifier biases: Neural collapse inspired federated learning with synthetic and fixed classifier," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2023, pp. 5296–5306.

[19] Y. Lyu, L. Wang, X. Zhang, Z. J. Sun, H. Su, J. Zhu, and L. Jing, "Overcoming recency bias of normalization statistics in continual learning: Balance and adaptation," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 36, Jan. 2023, pp. 25475–25494.

[20] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu, "Large scale incremental learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 374–382.

[21] M. Masana, X. Liu, B. Twardowski, M. Menta, A. D. Bagdanov, and J. Van De Weijer, "Class-incremental learning: Survey and performance evaluation on image classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 5, pp. 5513–5533, May 2023.

[22] Z. Mai, R. Li, H. Kim, and S. Sanner, "Supervised contrastive replay: Revisiting the nearest class mean classifier in online class-incremental continual learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2021, pp. 3584–3594.

[23] J. Kirkpatrick, R. Pascanu, N. C. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwińska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," *Proc. Nat. Acad. Sci. USA*, vol. 114, no. 13, pp. 3521–3526, Mar. 2017.

[24] G. Legate, L. Caccia, and E. Belilovsky, "Re-weighted softmax cross-entropy to control forgetting in federated learning," in *Proc. Conf. Lifelong Learn. Agents*, Jan. 2023, pp. 764–780.

[25] D. Caldarola, B. Caputo, and M. Ciccone, "Improving generalization in federated learning by seeking flat minima," in *Proc. Eur. Conf. Comput. Vis.*, Jan. 2022, pp. 654–672.

[26] Y. Ruan, X. Zhang, S.-C. Liang, and C. Joe-Wong, "Towards flexible device participation in federated learning," in *Proc. Int. Conf. Artif. Intell. Statist. (AISTATS)*, vol. 130, Apr. 2021, pp. 3403–3411.

[27] T. Weyand, A. Araujo, B. Cao, and J. Sim, "Google landmarks dataset v2—A large-scale benchmark for instance-level recognition and retrieval," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 2572–2581.

[28] G. Van Horn, O. Mac Aodha, Y. Song, Y. Cui, C. Sun, A. Shepard, H. Adam, P. Perona, and S. Belongie, "The iNaturalist species classification and detection dataset," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8769–8778.

[29] T.-M. H. Hsu, H. Qi, and M. A. Brown, "Federated visual classification with real-world data distribution," in *Proc. 16th Eur. Conf. Comput. Vis. (ECCV)*, Glasgow, U.K. Cham, Switzerland: Springer, Jan. 2020, pp. 76–92.

[30] M. Luo, F. Chen, D. Hu, Y. Zhang, J. Liang, and J. Feng, "No fear of heterogeneity: Classifier calibration for federated learning with non-IID data," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, Jan. 2021, pp. 5972–5984.

[31] E. Fanì, R. Camoriano, B. Caputo, and M. Ciccone, "Accelerating heterogeneous federated learning with closed-form classifiers," in *Proc. Int. Conf. Mach. Learn.*, 2024, pp. 1–7.

[32] K. Pillutla, K. Malik, A. Mohamed, M. Rabbat, M. Sanjabi, and L. Xiao, "Federated learning with partial model personalization," in *Proc. Int. Conf. Mach. Learn.*, Jan. 2022, pp. 17716–17758.

[33] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on non-IID data," in *Proc. Int. Conf. Learn. Represent.*, Jan. 2019, pp. 1–26.

[34] S. J. Reddi, A. Hefny, S. Sra, B. Poczos, and A. Smola, "Stochastic variance reduction for nonconvex optimization," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 314–323.

[35] H. C.-H. Hsu, H. Qi, and M. A. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," in *Proc. Neurips Workshop Federated Learn.*, Jan. 2019, pp. 1–5.

[36] J. Wang, V. Tantia, N. Ballas, and M. Rabbat, "SlowMo: Improving communication-efficient distributed SGD with slow momentum," in *Proc. Int. Conf. Learn. Represent.*, Jan. 2019, pp. 1–27.

[37] S. J. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečný, S. Kumar, and H. B. McMahan, "Adaptive federated optimization," in *Proc. Int. Conf. Learn. Represent.*, Jan. 2020, pp. 1–38.

[38] S. Praneeth Karimireddy, M. Jaggi, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. Theertha Suresh, "Mime: Mimicking centralized stochastic algorithms in federated learning," 2020, *arXiv:2008.03606*.

[39] R. Zaccone, C. Masone, and M. Ciccone, "Communication-efficient heterogeneous federated learning with generalized heavy-ball momentum," 2023, *arXiv:2311.18578*.

[40] Y. Yu, A. Wei, S. P. Karimireddy, Y. Ma, and M. I. Jordan, "TCT: Convexifying federated learning using bootstrapped neural tangent kernels," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, Jan. 2022, pp. 30882–30897.

[41] R. Wang, M. Ciccone, G. Luise, A. Yapp, M. Pontil, and C. Ciliberto, "Schedule-robust online continual learning," 2022, *arXiv:2210.05561*.

[42] B. Kang, S. Xie, M. Rohrbach, Z. Yan, A. Gordo, J. Feng, and Y. Kalantidis, "Decoupling representation and classifier for long-tailed recognition," in *Proc. 8th Int. Conf. Learn. Represent. (ICLR)*, Jan. 2019, pp. 1–16.

[43] B. Li, M. N. Schmidt, T. S. Alstrøm, and S. U. Stich, "On the effectiveness of partial variance reduction in federated learning with heterogeneous data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 3964–3973.

[44] G. Wan, W. Huang, and M. Ye, "Federated graph learning under domain shift with generalizable prototypes," in *Proc. AAAI Conf. Artif. Intell.*, vol. 38, Mar. 2024, pp. 15429–15437.

[45] J. Zhang, Y. Liu, H. Yang, and J. Cao, "FedTGP: Trainable global prototypes with adaptive-margin-enhanced contrastive learning for data and model heterogeneity in federated learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 38, Mar. 2024, pp. 16768–16776.

[46] W. Huang, Y. Liu, M. Ye, J. Chen, and B. Du, "Federated learning with long-tailed data via representation unification and classifier rectification," *IEEE Trans. Inf. Forensics Security*, vol. 19, pp. 5738–5750, 2024.

[47] G. Lee, M. Jeong, S. Kim, J. Oh, and S.-Y. Yun, "FedSOL: Stabilized orthogonal learning with proximal restrictions in federated learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2024, pp. 12512–12522.

[48] X. Shang, Y. Lu, G. Huang, and H. Wang, "Federated learning on heterogeneous and long-tailed data via classifier re-training with federated features," 2022, *arXiv:2204.13399*.

[49] J. Oh, S. Kim, and S.-Y. Yun, "FedBABU: Towards enhanced representation for federated image classification," in *Proc. Int. Conf. Learn. Represent.*, Jan. 2021, pp. 1–29.

[50] V. Papyan, X. Y. Han, and D. L. Donoho, "Prevalence of neural collapse during the terminal phase of deep learning training," *Proc. Nat. Acad. Sci. USA*, vol. 117, no. 40, pp. 24652–24663, Oct. 2020.

[51] D. Bacciu, D. Di Sarli, P. Faraji, C. Gallicchio, and A. Micheli, "Federated reservoir computing neural networks," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2021, pp. 1–7.

[52] G. Legate, N. Bernier, L. Caccia, E. Oyallon, and E. Belilovsky, "Guiding the last layer in federated learning with pre-trained models," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 36, Jan. 2023, pp. 69832–69848.

[53] A. Afonin and S. P. Karimireddy, "Towards model agnostic federated learning using knowledge distillation," in *Proc. Int. Conf. Learn. Represent.*, Jan. 2021, pp. 1–23.

[54] J. Cai, X. Liu, Z. Yu, K. Guo, and J. Li, "Efficient vertical federated learning method for ridge regression of large-scale samples," *IEEE Trans. Emerg. Topics Comput.*, vol. 11, no. 2, pp. 511–526, Jun. 2023.

[55] L. Huang, Z. Li, J. Sun, and H. Zhao, "Coresets for vertical federated learning: Regularized linear regression and k-means clustering," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 29566–29581.

[56] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–19, 2019.

[57] Y. Zhang, M. J. Wainwright, and J. C. Duchi, "Communication-efficient algorithms for statistical optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, 2012, pp. 1–8.

[58] Y. Zhang, J. Duchi, and M. Wainwright, "Divide and conquer kernel ridge regression: A distributed algorithm with minimax optimal rates," *J. Mach. Learn. Res.*, vol. 16, no. 1, pp. 3299–3340, 2015.

[59] P. Richtárik and M. Takáč, "Distributed coordinate descent method for learning with big data," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 2657–2681, 2016.

[60] T. Guo, S. Guo, J. Wang, X. Tang, and W. Xu, "PromptFL: Let federated participants cooperatively learn prompts instead of models—Federated learning in age of foundation model," *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 5179–5194, May 2024.

[61] J. Chen, W. Xu, S. Guo, J. Wang, J. Zhang, and H. Wang, "FedTune: A deep dive into efficient federated fine-tuning with pre-trained transformers," 2022, *arXiv:2211.08025*.

[62] J. Zhang, S. Vahidian, M. Kuo, C. Li, R. Zhang, T. Yu, G. Wang, and Y. Chen, "Towards building the federatedgpt: Federated instruction tuning," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2024, pp. 6915–6919.

[63] T. Zhang, T. Feng, S. Alam, D. Dimitriadis, S. Lee, M. Zhang, S. S. Narayanan, and S. Avestimehr, "GPT-FL: Generative pre-trained model-assisted federated learning," 2023, *arXiv:2306.02210*.

[64] S. Babakniya, A. Roushdy Elkordy, Y. H. Ezzeldin, Q. Liu, K.-B. Song, M. El-Khamy, and S. Avestimehr, "SLoRA: Federated parameter efficient fine-tuning of language models," 2023, *arXiv:2308.06522*.

[65] Y. Jee Cho, L. Liu, Z. Xu, A. Fahrezi, and G. Joshi, "Heterogeneous LoRa for federated fine-tuning of on-device foundation models," 2024, *arXiv:2401.06432*.

[66] L. Yi, H. Yu, G. Wang, X. Liu, and X. Li, "PFedLoRA: Model-heterogeneous personalized federated learning with LoRa tuning," 2023, *arXiv:2310.13283*.

[67] Y. Zhang and Q. Yang, "A survey on multi-task learning," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 12, pp. 5586–5609, Dec. 2022.

[68] M. Khodak, M.-F. Balcan, and A. Talwalkar, "Adaptive gradient-based meta-learning methods," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, Jan. 2019, pp. 5917–5928.

[69] F. Sattler, K.-R. Müller, and W. Samek, "Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 8, pp. 3710–3722, Aug. 2021.

[70] G. Huang, X. Chen, T. Ouyang, Q. Ma, L. Chen, and J. Zhang, "Collaboration in participant-centric federated learning: A game-theoretical perspective," *IEEE Trans. Mobile Comput.*, vol. 22, no. 11, pp. 6311–6326, Nov. 2022.

[71] M. Duan, D. Liu, X. Ji, R. Liu, L. Liang, X. Chen, and Y. Tan, "FedGroup: Efficient federated learning via decomposed similarity-based clustering," in *Proc. IEEE Int. Conf Parallel Distrib. Process. Appl., Big Data Cloud Comput., Sustain. Comput. Commun., Social Comput. Netw. (ISPA/BDCloud/SocialCom/SustainCom)*, Sep. 2021, pp. 228–237.

[72] D. Caldarola, M. Mancini, F. Galasso, M. Ciccone, E. Rodolà, and B. Caputo, "Cluster-driven graph federated learning over multiple domains," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2021, pp. 2743–2752.
[73] C. T. Dinh, N. Tran, and J. Nguyen, "Personalized federated learning with Moreau envelopes," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 21394–21405.
[74] T. Li, S. Hu, A. Beirami, and V. Smith, "Ditto: Fair and robust federated learning through personalization," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 6357–6368.
[75] Y. Deng, M. Mahdi Kamani, and M. Mahdavi, "Adaptive personalized federated learning," 2020, *arXiv:2003.13461*.
[76] T. Shen, J. Zhang, X. Jia, F. Zhang, G. Huang, P. Zhou, K. Kuang, F. Wu, and C. Wu, "Federated mutual learning," 2020, *arXiv:2006.16765*.
[77] F. Hanzely, B. Zhao, and M. Kolar, "Personalized federated learning: A unified framework and universal optimization techniques," 2021, *arXiv:2102.09743*.
[78] P. Pu Liang, T. Liu, L. Ziyin, N. B. Allen, R. P. Auerbach, D. Brent, R. Salakhutdinov, and L.-P. Morency, "Think locally, act globally: Federated learning with local and global representations," 2020, *arXiv:2001.01523*.
[79] K. Singhal, H. Sidahmed, Z. Garrett, S. Wu, J. Rush, and S. Prakash, "Federated reconstruction: Partially local federated learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 11220–11232.
[80] L. Collins, H. Hassani, A. Mokhtari, and S. Shakkottai, "Exploiting shared representations for personalized federated learning," in *Proc. Int. Conf. Mach. Learn.*, Jan. 2021, pp. 2089–2099.
[81] S. M. Stigler, "Gauss and the invention of least squares," *Ann. Statist.*, vol. 9, no. 3, pp. 465–474, May 1981.
[82] Å. Björck, *Numerical Methods for Least Squares Problems*. Philadelphia, PA, USA: SIAM, 1996.
[83] R. Rifkin, G. Yeo, and T. Poggio, "Regularized least-squares classification," *Nato Sci. Ser. Sub Comput. Syst. Sci.*, vol. 190, pp. 131–154, Jul. 2003.
[84] C. Bishop, *Pattern Recognition and Machine Learning*, vol. 2. Cham, Switzerland: Springer, 2006, pp. 531–537.
[85] S. P. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[86] I. Steinwart and A. Christmann, *Support Vector Machines*. Cham, Switzerland: Springer, 2008.
[87] P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe, "Convexity, classification, and risk bounds," *J. Amer. Stat. Assoc.*, vol. 101, no. 473, pp. 138–156, Mar. 2006.
[88] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[89] R. Camoriano, G. Pasquale, C. Ciliberto, L. Natale, L. Rosasco, and G. Metta, "Incremental robot learning of new objects with fixed update time," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 3207–3214.
[90] T. Kailath, A. H. Sayed, and B. Hassibi, *Linear Estimation*. Upper Saddle River, NJ, USA: Prentice-Hall, 2000.
[91] J. Sherman and W. J. Morrison, "Adjustment of an inverse matrix corresponding to a change in one element of a given matrix," *Ann. Math. Statist.*, vol. 21, no. 1, pp. 124–127, Mar. 1950.
[92] W. W. Hager, "Updating the inverse of a matrix," *SIAM Rev.*, vol. 31, no. 2, pp. 221–239, Jun. 1989.
[93] A. H. Sayed, *Adaptive Filters*. Hoboken, NJ, USA: Wiley, 2008.
[94] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 20, Dec. 2007, pp. 1177–1184.
[95] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.
[96] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.
[97] A. Krizhevsky, "Learning multiple layers of features from tiny images," Master's thesis, Univ. Toronto, Toronto, ON, Canada, 2009. [Online]. Available: https://www.cs.utoronto.ca/~kriz/learning-features-2009-TR.pdf
[98] D. M. Jimenez G., D. Solans, M. Heikkila, A. Vitaletti, N. Kourtellis, A. Anagnostopoulos, and I. Chatzigiannakis, "Non-IID data in federated learning: A survey with taxonomy, metrics, methods, frameworks and future directions," 2024, *arXiv:2411.12377*.
[99] G. Luo, N. Chen, J. He, B. Jin, Z. Zhang, and Y. Li, "Privacy-preserving clustering federated learning for non-IID data," *Future Gener. Comput. Syst.*, vol. 154, pp. 384–395, May 2024.
[100] J. Yoon, W. Jeong, G. Lee, E. Yang, and S. J. Hwang, "Federated continual learning with weighted inter-client transfer," in *Proc. Int. Conf. Mach. Learn.*, May 2021, pp. 12073–12086.
[101] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, vol. 2. Cham, Switzerland: Springer, 2009.
[102] Y. Tan, G. Long, L. Liu, T. Zhou, Q. Lu, J. Jiang, and C. Zhang, "FedProto: Federated prototype learning across heterogeneous clients," in *Proc. AAAI Conf. Artif. Intell.*, vol. 36, Jun. 2022, pp. 8432–8440.
[103] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava, "A survey on security and privacy of federated learning," *Future Gener. Comput. Syst.*, vol. 115, pp. 619–640, Oct. 2021.
[104] Q. Li, Z. Wen, Z. Wu, S. Hu, N. Wang, Y. Li, X. Liu, and B. He, "A survey on federated learning systems: Vision, hype and reality for data privacy and protection," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 4, pp. 3347–3366, Apr. 2023.
[105] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for federated learning on user-held data," 2016, *arXiv:1611.04482*.
[106] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of Cryptography*, S. Halevi and T. Rabin, Eds., Berlin, Germany: Springer, 2006, pp. 265–284.
[107] A. E. Ouadrhiri and A. Abdelhadi, "Differential privacy for deep and federated learning: A survey," *IEEE Access*, vol. 10, pp. 22359–22380, 2022.
[108] H. Fang and Q. Qian, "Privacy preserving machine learning with homomorphic encryption and federated learning," *Future Internet*, vol. 13, no. 4, p. 94, Apr. 2021.
[109] V. Mugunthan, A. Polychroniadou, D. Byrd, and T. H. Balch, "SMPAi: Secure multi-party computation for federated learning," in *Proc. NeurIPS Workshop Robust AI Financial Services*, vol. 21, Cambridge, MA, USA, 2019, pp. 1–17.
[110] Y. Yan and L. Zhu, "A simple data augmentation for feature distribution skewed federated learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jan. 2023, pp. 8074–8083.

EROS FANÌ received the B.Sc. degree in computer engineering in 2019 and the M.Sc. degree in data science and engineering in 2021 with a maximum score from the Polytechnic University of Turin. He is a European Laboratory for Learning and Intelligent Systems (ELLIS) Ph.D. candidate at the Polytechnic University of Turin, supervised by Prof. Barbara Caputo and co-advised by Dr. Marco Ciccone. He has recently concluded a visiting period at the University of Sussex, advised by Prof. Novi Quadrianto, and he is currently employed as a Research Fellow Software Engineer at the Basque Centre for Applied Mathematics. His research expertise and primary interests include federated learning and its real-world applications, such as semantic segmentation for autonomous driving, domain adaptation, and domain generalization. He received the Outstanding Reviewer Award at the European Conference on Computer Vision in 2024 (ECCV24).

RAFFAELLO CAMORIANO received the B.Sc., M.Sc., and Ph.D. degrees in computer and robotics engineering from the University of Genoa, Italy, in 2011, 2013, and 2017, respectively. He has been a Research Fellow and a Postdoctoral Researcher with the Istituto Italiano di Tecnologia, Genoa, from 2014 to 2022. He is currently an Assistant Professor with the Department of Control and Computer Engineering, Politecnico di Torino, Turin, Italy, and an Affiliated Researcher with the Istituto Italiano di Tecnologia. He is the author of 20 international peer-reviewed articles in the fields of efficient learning from structured data, continual learning, reinforcement learning, and robotics. He is a member of the European Laboratory for Learning and Intelligent Systems (ELLIS). He received the IEEE Computational Intelligence Society Italy Section Chapter's 2017 Best Ph.D. Thesis Award. He serves as an Associate Editor for IEEE Robotics and Automation Letters, *Machine Learning* (Springer), and IEEE/RSJ IROS.

BARBARA CAPUTO received the Ph.D. degree in computer science from the KTH Royal Institute of Technology, Stockholm, Sweden, in 2005. From 2007 to 2013, she was a Senior Researcher with Idiap-EPFL. Then, she moved to Sapienza Rome University, thanks to a MUR professorship, and joined the Politecnico di Torino, in 2018. She is currently a Full Professor with the Politecnico di Torino, where she leads the Artificial Intelligence (AI) Hub. Since 2017, she has been a double affiliation with the Italian Institute of Technology (IIT). She is one of the 30 experts who contributed to write the Italian strategy on AI and coordinator of the Italian National Ph.D. on AI and industry 4.0, sponsored by MUR. She is an ERC Laureate, ELLIS Fellow, and since 2019, she serves on the ELLIS Board.

MARCO CICCONE received the B.Sc. degree in computer engineering from the University of Florence, in 2013, and the M.Sc. degree in computer science and engineering and the Ph.D. degree in information technology from the Polytechnic University of Milan, in 2016 and 2021, respectively. During the Ph.D. study, he was a Research Intern with NVIDIA and NNAISENSE. From 2021 to 2024, he was an ELLIS Postdoctoral Researcher with the Polytechnic University of Turin and University College London. He is currently a Postdoctoral Fellow with the Vector Institute, Toronto, where he focuses on building modular machine learning systems that can efficiently learn to solve new problems by reusing, composing, and improving previously learned skills. This includes research on federated learning, to enable collaboration and decentralization by sharing knowledge and resources while preserving privacy; continual learning, to enable continuous knowledge integration without forgetting; modular learning, for designing more reusable, maintainable, and efficient models. He served as the Competition Track Co-Chair for NeurIPS in 2021, 2022, and 2023.