FeO2: Federated Learning with Opt-Out Differential Privacy

Nasser Aldaghri & Hessam Mahdavifar
University of Michigan
`{aldaghri,hessam}@umich.edu`

Ahmad Beirami
MIT
`beirami@mit.com`

Abstract

Federated learning (FL) is an emerging privacy-preserving paradigm, where a global model is trained at a central server while keeping client data local. However, FL can still indirectly leak private client information through model updates during training. Differential privacy (DP) can be employed to provide privacy guarantees within FL, typically at the cost of degraded final trained model. In this work, we consider a heterogeneous DP setup where clients are considered private by default, but some might choose to opt out of DP. We propose a new algorithm for federated learning with opt-out DP, referred to as *FeO2*, along with a discussion on its advantages compared to the baselines of private and personalized FL algorithms. We prove that the server-side and client-side procedures in *FeO2* are optimal for a simplified linear problem. We also analyze the incentive for opting out of DP in terms of performance gain. Through numerical experiments, we show that *FeO2* provides up to 9.27% performance gain in the global model compared to the baseline DP FL for the considered datasets. Additionally, we show a gap in the average performance of personalized models between non-private and private clients of up to 3.49%, empirically illustrating an incentive for clients to opt out.

1 Introduction

The abundance of data and advances in computation infrastructure have enabled the training of high-quality machine learning models. On the other hand, the data is distributed over many devices that are typically power-constrained and have limited computational capabilities. To reduce the amount of data transmission over networks and maintain the privacy of raw data, [1] proposed the federated learning (FL) framework for training a central server-side model using decentralized data at clients. See the recent surveys [2]; [3] for more. Federated learning frameworks aim to train a global model iteratively and collaboratively using clients' data. During each round, the server has access to a select number of clients, each of whom has a local dataset. The server broadcasts the current model to such clients, who train the model by taking gradient steps using their local data on the model and return the gradient-based update back to the server. The server then aggregates the updates and produces the new global model for the next round.

Despite the clients' data being kept on device in federated learning, the deployed model at the central server is still vulnerable to various privacy attacks, such as membership inference attacks, model inversion attacks, and others. To resolve this issue, many works in the literature propose privacy-preserving variations of federated learning algorithms. One approach to privacy-preserving FL algorithms utilizes differential privacy to provide privacy guarantees. Differential privacy is a widely studied and accepted mathematical notion that describes privacy-preserving algorithms where the information leakage of private data is bounded. Differential privacy is defined as follows

Definition 1 (differential privacy (DP) [4])**.** *A randomized algorithm $A(\cdot)$, whose image is denoted as $\boldsymbol{O}$, is said to be (ϵ, δ)-DP if for any two inputs $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$ that differ in just one entry, and all*

Extended version of the paper is available at `https://arxiv.org/abs/2110.15252` *subsets $O \subseteq \boldsymbol{O}$ the following relationship holds*

$$\Pr(A(\boldsymbol{D}_1) \in O) \leq e^{\epsilon} \Pr(A(\boldsymbol{D}_2) \in O) + \delta. \tag{1}$$

In federated learning, instead of targeting the privacy of individual samples of a client, one can apply client-level differential privacy by having the adjacent datasets describe the case where the data removed is all the samples associated with a single client [5]. It is worth noting that using differential privacy in federated learning causes unavoidable degradation in performance.

Although the trained global model in FL can perform well on average, it is not straightforward to guarantee the global model performs well for all clients on their local data, especially when considering heterogeneity in client data. This has encouraged many different works in the literature to obtain global models that work well for all clients or personalization techniques for federated learning setups to improve the performance of models at clients.

1.1 Contributions

This work examines heterogeneity in privacy requirements in federated learning setups. It considers a new setup of privacy-preserving federated learning where privacy is enabled by default and the clients may choose to opt out. We desire to understand the implications of the setting where a fraction of the clients choose to opt out of privacy, even if they represent a small percentage of the overall population, to improve the performance of the global model as well as the personalized local models. Specifically,

1. We propose a new setup for privacy in federated learning frameworks. The proposed setup considers heterogeneity in privacy choices of clients in FL. Instead of enabling privacy for all clients, each client is by default opted in privacy, and given the option to opt out, making it easier for clients to decide their privacy choices. We consider the client-level notion of privacy for the set of private clients rather than the entire set of clients.
2. To address this heterogeneity in privacy requirements, we propose the federated learning with opt-out differential privacy algorithm, referred to as *FeO2*. The *FeO2* algorithm is designed to take advantage of the newly introduced setup to improve the performance of the global model. *FeO2* comes with a personalization add-on, where we employ a recently-introduced algorithm, known as *Ditto* [6], within *FeO2* to examine the effect of *FeO2* on the local models.
3. We theoretically investigate *FeO2* in a simplified setup of federated point estimation, first introduced by [6]. We show that the *FeO2* algorithm can significantly improve the estimation noise variance at the server while providing better personalized estimates at clients. We also establish the Bayes optimality of *FeO2* when combined with *Ditto*.
4. Finally, we provide experimental results of the *FeO2* algorithm using various synthetic and realistic federated datasets from TensorFlow Federated (TFF) [7].

1.2 Related Work

Federated learning has been garnering a huge amount of efforts in the literature over the past few years. Works on federated learning algorithms have been proposed in the literature to overcome various issues that arise in realistic federated learning setups, e.g., [8, 9, 10, 11, 12, 13, 14, 15, 16, 17] to name a few. In federated learning, the data generated by clients is not considered to be independent and identically distributed (IID) as in distributed or centralized learning setups; therefore, the resulting model at the server may not necessarily perform well on the local data at each client. Personalization is a solution to this issue. By modifying algorithms to provide clients with better models that perform well on their local data, see for example, [18, 19, 20, 21, 22, 23, 24, 25, 6].

The trained models in federated learning are vulnerable to various privacy attacks. Differential privacy has been employed with federated learning in various works in the literature to provide privacy guarantees for such algorithms. For example, [26, 27, 28] apply DP mechanisms at clients to ensure sample-level DP guarantees. On the other hand, [29, 5, 30, 31] apply DP mechanisms at the server to ensure client-level DP guarantees. Applying DP typically causes some degradation in performance, i.e., the model's performance degrades as the privacy budget gets smaller. These approaches to privacy-preserving federated learning use fixed privacy budgets for all clients, an approach that can be overly strict and cause unnecessary degradation in performance. Our work is focused on giving clients the option to opt out and utilizing such information at the server to produce better models. A variation of DP setups is proposed in the literature, for scenarios with clients and a server, to use a hybrid model by combining sample-level DP with client-level DP and giving clients the option to opt in either [32, 33].

2 Privacy & Personalization In Federated Learning

The federated learning setup consists of a central server and a set of clients. The central server coordinates the training of the model using the clients over multiple training rounds. The set of all clients, denoted by $\mathcal{C}$, contains all clients that wish to cooperate in training the model. Each client $c_i \in \mathcal{C}$ has a local dataset, denoted by $\boldsymbol{D}_i$, and a local loss function $f_i(\cdot)$. During each communication round t, the server sends the current model state, i.e., $\boldsymbol{\theta}^t$, to the set of available clients in that round, denoted by $\mathcal{C}^t$, who train the model on their local datasets to minimize their local loss functions $f_i(\cdot)$. The clients then return the updated model to the server who aggregates them, e.g., averages them, to produce the next model state $\boldsymbol{\theta}^{t+1}$. Next, we describe an approach for privacy-preserving federated learning as well as an approach to personalized federated learning. The proposed algorithm *FeO2* employs a version of both approaches, and they are considered the two baselines against which *FeO2* is examined.

2.1 Privacy-Preserving Federated Learning: *DP-FedAvg*

To design privacy-preserving federated learning algorithms using differential privacy, a few modifications are required to the baseline federated averaging algorithm. Specifically, two modifications are introduced: clipping and noising. Considering client-level privacy, the averaging operation at the server is the target of such modifications. Assume clients are selected each round with probability q from the population of all clients of size N. First, each client update is clipped to have norm at most S, then the average is computed and additive Gaussian noise is added with mean zero and co-variance $\sigma^2 I = z^2(\frac{S}{qN})^2 I$. The variable z is referred to as the noise multiplier, which dictates the achievable values of (ϵ, δ)-DP. Training the model via multiple rounds increases the amount of leaked information, the moment accountant method, introduced by [34], can be used to provide a tighter estimate of the resulting DP parameters (ϵ, δ).

Selecting the clipping threshold as well as the noise multiplier is instrumental to getting useful models with desirable privacy guarantees. During training, the norm of updates can either increase or decrease, if the norm increases or decreases significantly compared to the clipping norm, the algorithm may slow down or diverge. Hence, [30] presented a solution to privately and adaptively update the clipping norm during each round of communication in federated learning based on the feedback from clients on whether or not their update norm exceeded the clipping norm. We consider this as the baseline for privacy-preserving federated learning algorithm and refer to it in the rest of the paper as *DP-FedAvg*. The case where no noise is added is the baseline for non-private federated learning algorithm, which is referred to simply as *non-private*.

2.2 Personalized Federated Learning: *Ditto*

There are multiple approaches to personalization in federated learning as mentioned in the previous section. In this paper, we consider a recent work by [6], referred to as *Ditto*, due to its simplicity and modularity. *Ditto* is a bi-level optimization objective, which does not modify the global FL training process, but adds a local training task to personalize a model locally with the following local loss function:

$$h_i(\boldsymbol{\theta}_i, \boldsymbol{\theta}^*) = f_i(\boldsymbol{\theta}_i) + \frac{\lambda_i}{2}\|\boldsymbol{\theta}_i - \boldsymbol{\theta}^*\|^2, \tag{2}$$

where $\boldsymbol{\theta}_i$ is the personalized local model for client c_i, λ_i is the regularization parameter for client c_i, and $\boldsymbol{\theta}^*$ is the global server-side model. As *Ditto* can simply be added onto the previously described learning algorithms, we combine it with them to train personalized local models for each client. This forms the baseline for the personalized learning in both *DP-FedAvg* and *non-private*.

3 Federated Learning with Opt-Out Differential Privacy: *FeO2*

In this section, we describe the *FeO2* algorithm that is designed to take advantage of the aforementioned heterogeneous privacy setup. The proposed algorithm is described in Algorithm 1. As mentioned before, the *FeO2* algorithm utilizes differential privacy with adaptive clipping, and optionally *Ditto* for personalized learning.

First, we provide the notation of the variables in the algorithm. We split the set of clients $\mathcal{C}$ into two subsets containing private and non-private clients denoted by $\mathcal{C}_{\text{p}}$ and $\mathcal{C}_{\text{np}}$, respectively. Assume the number of private and non-private clients is $N_{\text{p}} = (1-\rho_{\text{np}})N$ and $N_{\text{np}} = \rho_{\text{np}}N$, where ρ_{np} is the fraction of clients who opt out. The rest of the hyperparameters in the algorithm are as follows: the ratio hyperparameter r, the noise multipliers z, z_b, the clipping sensitivity S, the learning rate at clients η, the personalized learning rate at clients η_p, quantile κ, and factor η_b. The superscript $(\cdot)^t$ is used to denote a parameter during the t-th training round. During round t of training, no additional

Algorithm 1 FeO2: Federated learning with opt-out DP

```
*Inputs:* model parameters $\boldsymbol{\theta}^0$, sensitivity $S^0$, learning rate $\eta$, personalized learning rate $\eta_p$, ratio $r$, noise multipliers $z$, $z_b$, quantile $\kappa$, and factor $\eta_b$.
*Outputs:* $\boldsymbol{\theta}^T$, $\{\boldsymbol{\theta_j}\}_{j\in[N]}$
**At server:**
**for** round $t = 0, 1, 2, ..., T-1$ **do**
    $\mathcal{C}^t \leftarrow$ Sample $N^t$ clients from $\mathcal{C}$
    **for** client $c_j$ in $\mathcal{C}^t$ **in parallel do**
        $\triangle\boldsymbol{\theta}_j^t, b_j^t \leftarrow$ *ClientUpdate*($\boldsymbol{\theta}^t, c_j, S^t$)
    **end for**
    $N_{\text{p}}^t \leftarrow |\mathcal{C}_{\text{p}}^t|, \quad N_{\text{np}}^t \leftarrow |\mathcal{C}_{\text{np}}^t|, \quad z^t \leftarrow z\frac{S^t}{N_{\text{p}}^t}$
    $\triangle\boldsymbol{\theta}_{\text{p}}^{t+1} \leftarrow \frac{1}{N_{\text{p}}^t}\sum_{i\in\mathcal{C}_{\text{p}}^t}\triangle\boldsymbol{\theta}_i^t + \mathcal{N}(\mathbf{0}, (z^t)^2\mathbf{I})$
    $\triangle\boldsymbol{\theta}_{\text{np}}^{t+1} \leftarrow \frac{1}{N_{\text{np}}^t}\sum_{i\in\mathcal{C}_{\text{np}}^t}\triangle\boldsymbol{\theta}_i^t$
    $\boldsymbol{\theta}^{t+1} \leftarrow \boldsymbol{\theta}^t + \frac{1}{N_{\text{np}}^t + rN_{\text{p}}^t}[N_{\text{np}}^t \triangle \boldsymbol{\theta}_{\text{np}}^{t+1} + rN_{\text{p}}^t \triangle \boldsymbol{\theta}_{\text{p}}^{t+1}]$
    $S^{t+1} \leftarrow S^t e^{-\eta_b\left(\left(\frac{1}{N^t}\sum_{i\in\mathcal{C}^t} b_i^t + \mathcal{N}(0, z_b^2 \frac{1}{N^t}^2)\right) - \kappa\right)}$
**end for**

**At client $c_j$:**
*ClientUpdate*($\boldsymbol{\theta}^0, c_j, S$):
$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^0$
$\boldsymbol{\theta}_j \leftarrow \boldsymbol{\theta}^0$ (if not initialized)
$\mathcal{B} \leftarrow$ batch the client's data $\boldsymbol{D}_j$
**for** epoch $e = 1, 2, ..., E$ **do**
    **for** $B$ in $\mathcal{B}$ **do**
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta\nabla f_j(\boldsymbol{\theta}, B)$
        $\boldsymbol{\theta}_j \leftarrow \boldsymbol{\theta}_j - \eta_p(\nabla f_j(\boldsymbol{\theta}_j, B) + \lambda(\boldsymbol{\theta}_j - \boldsymbol{\theta}^0))$. [*Ditto: optional for personalization*]
    **end for**
**end for**
$\triangle\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \boldsymbol{\theta}^0$
$b \leftarrow \mathbb{1}_{\|\triangle\boldsymbol{\theta}\|_2 \leq S}$
return Clip($\triangle\boldsymbol{\theta}, S$), $b$ to server

*Clip*($\boldsymbol{\theta}, S$):
    return $\boldsymbol{\theta} \times \frac{S}{\max(\|\boldsymbol{\theta}\|_2, S)}$ to client
```

steps are required for the clients during model training. Clients train the received model using their local data then send back their clipped updates $\Delta\boldsymbol{\theta}_j^t$ along with their clipping indicator b_j^t to the server. The server collects the updates from clients and performs a two-step aggregation process to the updates. During the first step, the updates from the non-private clients are passed through an averaging function to produce $\triangle\boldsymbol{\theta}_{\text{np}}^{t+1}$, while the updates from the private clients are passed through a differentially private-averaging function to produce $\triangle\boldsymbol{\theta}_{\text{p}}^{t+1}$. The second step of the aggregation is combining the outputs of the previous two averaging functions to produce the next iteration of the model. In this step, the server performs a weighted average of the two outputs. The weights for each are chosen based on the number of private and non-private clients, as well as a *ratio* hyperparameter r. The output of this step is $\triangle\boldsymbol{\theta}^{t+1}$, which is then added to the previous model state to produce the next model state.

The ratio hyperparameter r is chosen based on multiple factors, which we discuss next. In general, we desire the value of r be bounded as $0 \leq r \leq 1$ in *FeO2* to use non-private clients' updates more meaningfully. The first factor to consider when choosing r is related to the desired privacy budget, lower privacy budget ϵ requires more noise to be added, leading to a lower value of r. This intuition follows from observing Lemma 11 in [35] as will be shown in a simplified example in the next section. Another factor that is more difficult to quantify is the heterogeneity between the non-private set of clients and the private set of clients, more heterogeneity leads to higher values of r and vice versa.

As for the personalization part, we have the hyperparameter λ. *FeO2* differs from *Ditto* in a number of major ways. First, The server-side aggregation in *Ditto* is the vanilla *FedAvg*; however, in *FeO2* the server-side aggregation is not *FedAvg*, but rather a new aggregation rule which utilizes the privacy choices made by clients. Second, *Ditto* is designed for robustness against malicious clients; hence, the performance on malicious clients is not measured. That is not the case in *FeO2*, where measuring the performance for private and non-private clients is needed, and improving both is desired. Third, the server in *Ditto* is unaware of the status of the clients, i.e., whether or not they are malicious; while in *FeO2* the server is aware of the privacy choices made by clients which can be used during training.

4 Analyzing Federated Point Estimation

In this section, we provide some insights into the proposed *FeO2* algorithm through a simplified setup, inspired by the one proposed by [6], known as federated point estimation. We first start by considering the global estimation on the server and show that *FeO2* is Bayes optimal when using the appropriate value of the ratio r. Then we consider personalization using *Ditto* for both private and non-private clients and show that combining the optimal *FeO2* with *Ditto* is Bayes optimal for private and non-private clients when using appropriate values of the regularization parameters λ_{p} and λ_{np}. Finally, we show the gain of opting out in the proposed personalized setup compared to the baseline. The federated point estimation is presented for clarity of discussion, refer to the extended version for additional discussions and an extension of the setup to linear regression problems.

4.1 Setup

In this simplified setup, we consider a single round of communication and assume that all clients have the same number of samples, and that the effect of clipping is negligible. Let n_s denote the number of samples held by each client. Also, let us denote the point to be estimated at client c_j as $\phi_j = \phi + p_j$, where ϕ is the parameter to be estimated at the server, $p_j \sim \mathcal{N}(0, \tau^2)$ is the inherent Gaussian noise, which encompasses the non-IID nature in federated learning setups we are interested in. Increasing τ^2 makes the points more unrelated at different clients, and setting $\tau^2 = 0$ denotes the case of IID clients. The observed samples at client c_j are denoted by $\boldsymbol{x}_j = \{x_{j,1}, x_{j,2}, ..., x_{j,n_s}\}$, where $x_{j,i} = \phi_j + v_{j,i}$, where $v_{j,i} \sim \mathcal{N}(0, \beta^2)$ is the additive noise in the observations. The loss function at the client is $f_j(\phi) = \frac{1}{2}\left(\phi - \frac{1}{n_s}\sum_{i=1}^{n_s} x_{j,i}\right)^2$.

Let $\alpha^2 = \frac{\beta^2}{n_s}$. Then minimizing $f_j(\phi)$ leads the client to have the estimate $\hat{\phi}_j = \frac{1}{n_s}\sum_{i=1}^{n_s} x_{j,i}$, whose variance is $\sigma_c^2 = \alpha^2 + \tau^2$. For simplicity and clarity, we move the noise addition process from the server-side to the client side such that when the server aggregates the private clients' updates the resulting privacy noise variance is equivalent to the desired value by the server. Note that the notion of privacy here is still client-level privacy. We denote the updates sent to the server by client c_j as $\psi_j = \hat{\phi}_j + l_j$, where $l_j = 0$ for non-private clients and $l_j \sim \mathcal{N}(0, N_p\gamma^2)$ for private clients. Also, $\gamma^2 \propto \frac{1}{N_{\text{p}}^2}$ is the desired privacy noise variance at the server, which is related to the value of z^2 in Algorithm 1. In this setup, the server and clients goals is to minimize the Bayes risk (i.e., test error), defined as follows:

$$\theta^* := \arg\min_{\widehat{\theta}} \left\{ \mathbb{E}\left[\frac{1}{2}\left(\phi - \widehat{\theta}\right)^2 \middle| \psi_1, ..., \psi_N \right] \right\}. \tag{Glob. obj.}$$

$$\theta_j^* := \arg\min_{\widehat{\theta}} \left\{ \mathbb{E}\left[\frac{1}{2}\left(\phi_j - \widehat{\theta}\right)^2 \middle| \{\psi_i : i \in [N]\backslash j\}, \hat{\phi}_j \right] \right\}. \tag{Loc. obj.}$$

4.2 Global Model On The Server

The server's goal is to combine the updates received from clients such that the resulting noise variance is minimized, while ensuring the privacy of the set of private clients. To this end, we use Lemma 11 in [35] to find the optimal aggregation at the server. The server first computes the two intermediate average values for non-private and private clients as $\theta_{\text{np}} = \frac{1}{N_{\text{np}}}\sum_{i\in\mathcal{C}_{\text{np}}} \psi_i$, and $\theta_{\text{p}} = \frac{1}{N_{\text{p}}}\sum_{i\in\mathcal{C}_{\text{p}}} \psi_i$, where $\theta_{\text{np}} \sim \mathcal{N}(\phi, \frac{1}{N_{\text{np}}}\sigma_c^2)$, and $\theta_{\text{p}} \sim \mathcal{N}(\phi, \frac{1}{N_{\text{p}}}\sigma_c^2 + \gamma^2)$. Now, the server aims to combine such values to compute its estimation θ of the value of ϕ with the goal of minimizing the resulting estimation noise variance σ_s^2.

Lemma 1 (Global estimate optimality)**.** *FeO2 from the server's point of view, with ratio* $r^* = \frac{\sigma_c^2}{\sigma_c^2 + N_p\gamma^2}$*, is Bayes optimal (i.e.,* θ *converges to* θ^**) in the considered federated point estimation problem. Furthermore, the resulting variance is* $\sigma_{s,\text{opt}}^2 = \frac{1}{N}\left[\frac{\sigma_c^2(\sigma_c^2 + N_p\gamma^2)}{\sigma_c^2 + \rho_{np}N_p\gamma^2}\right]$.

4.3 Personalized Local Models On Clients

In this part, we consider using *Ditto* to train personalized local models at clients. We show that combining *Ditto* with *FeO2* provides the Bayes optimal local model for clients. We first note that in vanilla *Ditto*, the server applies vanilla *FedAvg* to produce the global model, while in *FeO2* with *Ditto* the server utilizes the side-information available to it about the privacy choices of clients to optimally aggregates their updates. Due to the server aggregation being different in these two algorithms, the resulting optimum λ's will also be different for clients in each algorithm. Additionally, as *FeO2* aims to provide optimal performance for both private and non-private clients, we need to find the optimal values of both λ_{np} and λ_{p}. Next, we discuss the optimality of *Ditto* combined with *FeO2* using the estimate θ^* for both private and non-private clients in the federated point estimation problem. To this end, we have the following lemma.

Lemma 2 (Personalized local estimate optimality)**.** *Assuming using Ditto combined with FeO2 with ratio* r^* *in Lemma 1, there exist* λ_{np}^* *for non-private clients and* λ_p^* *for private clients such that FeO2 is Bayes optimal (i.e.,* θ_j *converges to* θ_j^* *for each client* $j \in [N]$*).*

Expressions of λ_{np}^* and λ_{p}^* along with the corresponding proofs are provided in the extended version.

4.4 Incentive for Opting Out of DP

Earlier in this section, we have shown that for the problem of federated point estimation, the global estimate benefited greatly from the setup of opt-out differential privacy. A better global estimate enables better performance on clients' devices in the federated point estimation setup. However, a question may arise on whether or not clients are actually going to opt out of privacy. In other words, what incentivizes clients to opt out other than helping the server produce better estimates?

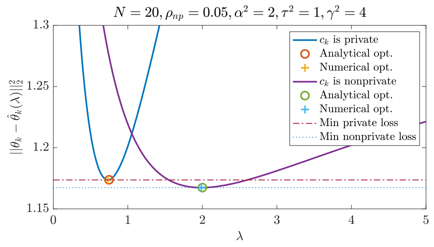

Figure 1: The effect of opting out on the personalized local model estimate as a function of λ.

Table 1: Summarized results of experiments on *synthetic datasets*: We compare the performance of the baseline algorithms against FeO2 with the hyperparameters that perform best. The variance of the performance metric across clients is between parenthesis.

nonIID MNIST dataset, $(3.6, 10^{-4})$-DP								
Setup		Global model				Personalized local models		
Algorithm	**hyperparameters**	$Acc_g\%$	$Acc_{\text{g,p}}\%$	$Acc_{\text{g,np}}\%$	$\triangle_{\text{g}}\%$	$Acc_{\text{l,p}}\%$	$Acc_{\text{l,np}}\%$	$\triangle_{\text{p}}\%$
Non-private	$\lambda_{\text{np}} = 0.005$	93.8	-	$93.75(0.13)$	-	-	$99.98(0.001)$	-
DP-FedAvg	$\lambda_{\text{p}} = 0.005$	88.75	$88.64(0.39)$	-	-	$99.97(0.002)$	-	-
FeO2	$r = 0.01$, $\lambda_{\text{p}} = \lambda_{\text{np}} = 0.005$	92.48	$92.43(0.30)$	$93.30(0.21)$	0.88	$99.94(0.001)$	$99.94(0.001)$	0.0
Skewed nonIID MNIST dataset, $(3.6, 10^{-4})$-DP								
Non-private	$\lambda_{\text{np}} = 0.005$	93.67	-	$93.62(0.15)$	-	-	$99.98(0.001)$	-
DP-FedAvg	$\lambda_{\text{p}} = 0.005$	88.93	$88.87(0.35)$	-	-	$99.98(0.001)$	-	-
FeO2	$r = 0.1$, $\lambda_{\text{p}} = \lambda_{\text{np}} = 0.005$	90.36	$89.96(0.37)$	$97.45(0.01)$	7.49	$99.97(0.001)$	$99.76(0.003)$	-0.21
FeO2	$r = 0.9$, $\lambda_{\text{p}} = \lambda_{\text{np}} = 0.005$	87.96	$87.69(0.56)$	$92.97(0.04)$	5.28	$99.98(0.001)$	$99.96(0.001)$	-0.02

To answer this question, we argue that opting out helps the server produce better global estimates for clients, in addition to helping clients produce better personalized local estimates. In other words, clients that opt out can produce better personalized local estimates compared to the ones that remain private. To illustrate the *motivation* of opting out for clients, we perform an experiment where we conduct the federated point estimation experiment for two scenarios. The first is the case where client c_k remains private, and the second is the case where c_k opts out of privacy and becomes non-private. The results of these experiments are shown in Figure 1. This illustrates the incentive for clients to opt out of privacy.

5 Experimental Evaluation

In this section, we present the results of a number of experiments to show the gain in performance of the proposed *FeO2* algorithm compared to the baseline *DP-FedAvg* algorithm. The experiments show that *FeO2* outperforms *DP-FedAvg* with the right choice of the hyperparameter r in terms of the global model accuracy, as well as in terms of the average personalized local model accuracy.

5.1 Setup

The experiments are conducted on multiple federated datasets, synthetic and realistic. The synthetic datasets are manually created to simulate extreme cases of data heterogeneity often exhibited in federated learning scenarios. The realistic federated datasets are from TFF [7], where such datasets are assigned to clients according to some criteria. The synthetic dataset is referred to as the non-IID MNIST dataset, and the number of samples at a client is fixed across all clients. Each client is assigned samples randomly from the subsets of samples each with a single digit between $0-9$. A skewed version of the synthetic dataset is one where non-private clients are sampled from the clients who have the digit 7 in their data. In the non-IID MNIST dataset, we have $2,000$ clients and we randomly sample 5% of them for training each round. The realistic federated datasets are the FMNIST and FEMNIST from TFF datasets. The FMNIST and FEMNIST datasets contain $3,383$ and $3,400$ clients, respectively, and we sample 3% of them for training each round. TensorFlow Privacy (TFP) [36] is used to compute the privacy loss incurred during training. Refer to the extended version for more extended description of the used models and extended results.

5.2 Results

In this part, we provide the outcomes of the experiments on the datasets mentioned above. In these experiments, we provide results for an opt-out rate of 5% of the total client population. Clients that opt out are picked randomly from the set of all clients but fixed for a fair comparison across all experiments. The exception for this assumption is for the skewed non-IID MNIST dataset, where clients that opt out are sampled from the clients who have the digit 7. All other hyperparameters are

Table 2: Summarized results of experiments on *realistic federated datasets*: We compare the performance of the baseline algorithms against FeO2 with the hyperparameters that perform best. The variance of the performance metric across clients is between parenthesis.

FMNIST dataset, $(0.6, 10^{-4})$-DP								
Setup		Global model				Personalized local models		
Algorithm	**hyperparameters**	$Acc_g\%$	$Acc_{\text{g,p}}\%$	$Acc_{\text{g,np}}\%$	$\triangle_{\text{g}}\%$	$Acc_{\text{l,p}}\%$	$Acc_{\text{l,np}}\%$	$\triangle_{\text{p}}\%$
Non-private	$\lambda_{\text{np}} = 0.05$	89.65	-	$89.35(1.68)$	-	-	$94.53(0.59)$	-
DP-FedAvg	$\lambda_{\text{p}} = 0.05$	77.61	$77.62(2.55)$	-	-	$90.04(1.04)$	-	-
FeO2	$r = 0.01$, $\lambda_{\text{p}} = 0.05, \lambda_{\text{np}} = 0.005$	86.88	$85.36(1.89)$	$90.02(1.28)$	4.66	$93.76(0.68)$	$95.94(0.41)$	2.18
FEMNIST dataset, $(4.1, 10^{-4})$-DP								
Non-private	$\lambda_{\text{np}} = 0.25$	81.66	-	$81.79(1.38)$	-	-	$84.46(0.89)$	-
DP-FedAvg	$\lambda_{\text{p}} = 0.05$	75.42	$75.86(1.82)$	-	-	$74.69(1.29)$	-	-
FeO2	$r = 0.1$, $\lambda_{\text{p}} = \lambda_{\text{np}} = 0.05$	76.52	$77.91(1.67)$	$83.9(1.27)$	5.99	$77.9(1.22)$	$79.15(0.99)$	1.25
FeO2	$r = 0.01$, $\lambda_{\text{p}} = \lambda_{\text{np}} = 0.25$	74.86	$77.31(2.18)$	$86.73(0.98)$	9.42	$81.19(1.02)$	$84.68(0.78)$	3.49

fixed. To evaluate the performance of each algorithm, we measure the following quantities for each dataset:

- Acc_{g}: the average test accuracy on the *server* test dataset using the global model.
- $Acc_{\text{g,p}}$, $Acc_{\text{g,np}}$: the average test accuracy of all *private* and *non-private* clients using the global model on their local test datasets, respectively.
- $Acc_{\text{l,p}}$, $Acc_{\text{l,np}}$: the average test accuracy of all *private* and *non-private* clients using their personalized local models on their local test datasets, respectively.
- $\triangle_{\text{g}}$, $\triangle_{\text{l}}$: the gain in the average performance of *non-private* clients over the *private* ones using the global model and the personalized local models on their local test datasets, respectively; computed as $\triangle_{\text{g}} = Acc_{\text{g,np}} - Acc_{\text{g,p}}$ and $\triangle_{\text{l}} = Acc_{\text{l,np}} - Acc_{\text{l,p}}$.

A summary of the results, shown in Table 1 and Table 2, provides the best performance for each experiment along with their corresponding hyperparameters. More detailed results are shown in the extended version. If different values of the hyperparameters in *FeO2* yield two different competing results, such as one with better global model performance at the server and one with better personalized local models at the clients, we show both.

We can see from Tables 1 and 2 that *FeO2* allows the server to learn better global models while allowing clients to learn better personalized local models compared to the baseline private FL, i.e., *DP-FedAvg*. For example, the gain due to *FeO2* compared to the *DP-FedAvg* in terms of global model performance is 3.8% on average and is up to 9.27%. For personalized local models, the gain in average accuracy for clients due to *FeO2* compared to *DP-FedAvg* is up to 9.99%. The gap in the average performance of personalized local models between *DP-FedAvg* and *non-private* is 3.57%, which is reduced to 0.95% between *FeO2* and *non-private*. Additionally, we can also see the gain in the average performance in personalized local models between clients who choose to opt out of privacy and clients who choose to remain private. This demonstrates the advantage of opting out, which provides clients with an incentive to opt out of differential privacy to improve their personalized local models, for example, non-private clients can gain up to 3.49% on average in terms of personalized local model performance compared to private clients. It is worth mentioning that opting out can also improve the global model's performance on clients' local data. We observe that there is up to 12.4% gain in the average performance of non-private clients in terms of the accuracy of the global model on the local data compared to the one of baseline *DP-FedAvg*.

6 Conclusion

In this paper, we considered a new aspect of heterogeneity in federated learning setups, namely heterogeneity in privacy requirements. We proposed a new setup for privacy heterogeneity between clients where privacy is no longer necessary for all clients, and some clients choose to opt out of privacy. We proposed a new algorithm called *FeO2* for the considered setup. In *FeO2*, the aim is to employ differential privacy to maintain the privacy of clients who choose to remain private, and we proposed a two-step aggregation scheme at the server to improve utility. Additionally, we employed *Ditto* as a personalization scheme to examine whether *FeO2* enhances the performance for personalized FL. We provided a treatment for the federated point estimation problem and showed the success of *FeO2* on the central server as well as the personalized local models. Finally, we provided a set of experiments on synthetic and realistic federated datasets and showed that *FeO2* outperforms the baseline private FL algorithm in terms of the global model as well as the personalized local models performance, and showed the incentive of becoming non-private compared to remaining private in such scenarios in terms of the gain in the average performance.

References

[1] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

[2] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

[3] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

[4] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.

[5] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. In *International Conference on Learning Representations*, 2018.

[6] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through personalization. *International Conference on Machine Learning*, 2021.

[7] Google. TensorFlow Federated, 2019.

[8] Jakub Konečný, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016.

[9] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[10] Luca Corinzia, Ami Beuret, and Joachim M Buhmann. Variational federated multi-task learning. *arXiv preprint arXiv:1906.06268*, 2019.

[11] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018.

[12] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

[13] Hangyu Zhu and Yaochu Jin. Multi-objective evolutionary federated learning. *IEEE transactions on neural networks and learning systems*, 31(4):1310–1322, 2019.

[14] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020.

[15] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and H Brendan McMahan. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*, 2020.

[16] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. *arXiv preprint arXiv:2002.06440*, 2020.

[17] Maruan Al-Shedivat, Jennifer Gillenwater, Eric Xing, and Afshin Rostamizadeh. Federated learning via posterior averaging: A new perspective and practical algorithms. *arXiv preprint arXiv:2010.05273*, 2020.

[18] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet Talwalkar. Federated multi-task learning. *arXiv preprint arXiv:1705.10467*, 2017.

[19] Kangkang Wang, Rajiv Mathews, Chloé Kiddon, Hubert Eichner, Françoise Beaufays, and Daniel Ramage. Federated evaluation of on-device personalization. *arXiv preprint arXiv:1910.10252*, 2019.

[20] Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. *arXiv preprint arXiv:1912.00818*, 2019.

[21] Mikhail Khodak, Maria-Florina Balcan, and Ameet Talwalkar. Adaptive gradient-based meta-learning methods. *arXiv preprint arXiv:1906.02717*, 2019.

[22] Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. Three approaches for personalization with applications to federated learning. *arXiv preprint arXiv:2002.10619*, 2020.

[23] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning: A meta-learning approach. *arXiv preprint arXiv:2002.07948*, 2020.

[24] Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. Adaptive personalized federated learning. *arXiv preprint arXiv:2003.13461*, 2020.

[25] Canh T Dinh, Nguyen H Tran, and Tuan Dung Nguyen. Personalized federated learning with moreau envelopes. *arXiv preprint arXiv:2006.08848*, 2020.

[26] Stacey Truex, Ling Liu, Ka-Ho Chow, Mehmet Emre Gursoy, and Wenqi Wei. LDP-Fed: Federated learning with local differential privacy. In *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking*, pages 61–66, 2020.

[27] Lichao Sun, Jianwei Qian, Xun Chen, and Philip S Yu. LDP-FL: Practical private aggregation in federated learning with local differential privacy. *arXiv preprint arXiv:2007.15789*, 2020.

[28] Muah Kim, Onur Günlü, and Rafael F Schaefer. Federated learning with local differential privacy: Trade-offs between privacy, utility, and communication. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2650–2654. IEEE, 2021.

[29] Robin C Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017.

[30] Galen Andrew, Om Thakkar, H Brendan McMahan, and Swaroop Ramaswamy. Differentially private learning with adaptive clipping. *arXiv preprint arXiv:1905.03871*, 2019.

[31] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469, 2020.

[32] Brendan Avent, Aleksandra Korolova, David Zeber, Torgeir Hovden, and Benjamin Livshits. BLENDER: Enabling local search with a hybrid differential privacy model. In *26th {USENIX} Security Symposium ({USENIX} Security 17)*, pages 747–764, 2017.

[33] Amos Beimel, Aleksandra Korolova, Kobbi Nissim, Or Sheffet, and Uri Stemmer. The power of synergy in differential privacy: Combining a small curator with local randomizers. *arXiv preprint arXiv:1912.08951*, 2019.

[34] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

[35] Hessam Mahdavifar, Ahmad Beirami, Behrouz Touri, and Jeff S Shamma. Global games with noisy information sharing. *IEEE Transactions on Signal and Information Processing over Networks*, 4(3):497–509, 2018.

[36] Google. TensorFlow Privacy, 2018.