Achieving Flexible Local Differential Privacy in Federated Learning via Influence Functions

Alycia N. Carey and Xintao Wu ✉

University of Arkansas, Fayetteville AR 72701, USA {ancarey, xintaowu}@uark.edu

Abstract. The use of local differential privacy in federated learning has recently grown in popularity due to rising demands for increased privacy in machine learning scenarios. While research into local differentially private federated learning is vast, the ability for a client to change their privacy parameter ε *after* training, and have that change reflected in the global model's parameters without having to repeat the entire federated training process, is currently unexplored. In this work, we propose ***FLDP-FL*** (Flexible Local Differential Privacy for Federated Learning), a simple and efficient technique for federated learning based on influence functions that enables clients to update their privacy guarantees after training without incurring extra training overhead by either the global server or the other federation participants. We show that our influence-based approach is able to accurately estimate the change in global model parameters that would occur if the client re-randomized their data under a stricter ε and the federated learning process was repeated. Additionally, we show that our FLDP-FL approach is able to reasonably estimate the resulting change when multiple clients update their privacy parameter ε.

Keywords: Federated Learning · Differential Privacy · Influence Functions.

1 Introduction

In federated learning, multiple clients jointly train a machine learning model under the orchestration of a central server without having to explicitly share their private local data with either the server or the other participants [16]. Despite being ingrained with an innate sense of privacy due to the clients' data remaining decentralized, the use of local differential privacy in federated scenarios has been widely considered to increase the overall privacy of the federated learning system. However, the solutions that have been proposed are often rigid, forcing clients to use the same privacy parameter ε and do not provide solutions in the event that a client has to update their ε in order to comply with updated privacy regulations. Today's privacy needs are ever changing, and thus it is important to construct federated learning architectures that are adaptable.

Granting clients the ability to post-hoc change their privacy parameters is a non-trivial task. For each instance of a client asking to change their ε, the federated model would have to be retrained from scratch which is time and resource intensive for *all* parties in the federation – not just the client changing ε. To costly federated retraining, while still offering clients flexibility in updating their privacy level as needed, we propose *flexible local differential privacy for federated learning* (FLDP-FL). We formulate FLDP-FL as an influence function [4] that is able to properly estimate the true model parameters that would be obtained if a client's data was re-randomized under a stronger ε and the federated model was fully retrained. Further, we do so without violating any federated learning requirements as the clients do not have to share their private data with the other clients or the global server to use FLDP-FL.

Our contributions are as follows: 1) We propose FLDP-FL, which is based on influence functions, to offer clients the ability to change their privacy level ε after federated training has concluded without requiring the entire federated training process to be repeated; 2) We extend previous influence function work and provide a lemma which shows that influence estimates for perturbing a data point are additive and then, based on the lemma, propose a theory for estimating the impact of a client modifying their ε value on the global model's parameters; and 3) We empirically show that FLDP-FL is able to properly estimate the model that would be obtained if federated training was re-performed under the client's updated ε. We additionally show that FLDP-FL can support multiple clients altering their privacy parameter post-hoc,. We note, however, that even though FLDP-FL supports the setting where the client changing ε owns a large portion of the federated dataset, the estimation can degrade beyond a reasonable amount and in these settings full retraining may be required.

2 Related Works

Differentially Private Federated Learning Differential privacy has been widely studied in the federated learning setting from the lens of "user-level" security and multiple approaches to ensuring differentially private federated learning have been proposed [2,7,8,17,24,26]. One of the first works in differentially private federated learning was [17] which proposed a noised version of the traditional federated averaging algorithm (FedAvg, [16]) which satisfies user-adjacent differential privacy via use of the moments accountant [1] and works by clipping the gradient updates of each user before they are sent by the client to the server, and then adding Gaussian noise to the averaged update. Other important differentially private federated learning works include [21] which proposed a local differential privacy-based parameter aggregation scheme and [8] which proposed a differentially private approach to crafting personalized models in federated learning. However, to our knowledge, no previous work has performed research into how a client can update their privacy parameter ε after federated training has concluded without having to repeat the entire federated training process.

Influence Functions and Federated Learning Influence functions are a product of influence analysis from the field of robust statistics [4] and were made popular in machine learning by [13] which showed how a single training point influences the final machine learning model's parameters and/or the test loss of a single test point. This work was further extended in [14] where the authors showed how influence functions can be used to estimate the influence that a group of training points has on the model parameters and/or the loss of a single test point. While both influence functions and differential privacy have strong ties to robust statistics [6], to our knowledge only two works have been published utilizing both methods [3,11] and they are both formulated in the centralized setting, not federated. However, work utilizing influence functions in federated learning have been proposed [9,15,18,19], such as [18] which proposed to filter and score data used in federated learning according to the sign of the influence function, [15] which proposed an influence-based participant selection strategy to mitigate test error caused by erroneous training data, and [19] which proposed an adaptive aggregation scheme based on class-level and client-level influence scores. To our knowledge, our work is the first to use influence functions to enable clients to change their local differential privacy parameter ε after federated training.

3 Preliminaries

In this section, we present the required background information on federated learning (FL), local differential privacy (LDP), and influence functions (IFs) needed to understand the discussions of Section 4. We begin by detailing the notation used through the remainder of the paper. We consider a federated learning system of N clients (denoted by i), each of which have a local training set $\mathcal{D}_{i,tr}$ made up of data with features $x_i \in \mathcal{X}_i$ and a label $y_i \in \mathcal{Y}_i$ such that $z_i = (x_i \in \mathcal{X}_i, y_i \in \mathcal{Y}_i) \in \mathcal{D}_{i,tr}$ is one of n_i training points belonging to client i. Each client has a local model $H_i(\theta)$, where $\theta \in \Theta$ represents the model parameters that are shared across the clients during training and we use $\hat{\theta}$ to denote the global model parameters obtained at the end of federated training. We assume that the global model has access to a testing set $\mathcal{D}_{te}$ that is a mixture of all N client's data distributions. We use $\ell(z_i, \theta) = \ell(H_i(x_i; \theta), y_i)$ to denote the loss function and $\mathcal{L}(\mathcal{D}_{i,tr}, \theta) = \frac{1}{n_i}\sum_{j=1}^{n_i} \ell(z_{i,j}, \theta)$ to denote the empirical risk and we assume that the empirical risk is twice-differentiable and strictly convex in θ for all clients [13]. Additionally, we use the standard notation from differential privacy of ε as the privacy parameter and $\mathbb{P}[\cdot]$ to denote probability.

3.1 Federated Learning

Federated learning is a machine learning setting where multiple clients (e.g., mobile devices, whole organizations, or individuals) collaboratively train a machine learning model under the orchestration of a central server, while keeping the training data decentralized. The most popular federated learning algorithm is Federated Averaging (FedAvg) [16] which aims to solve:

$$\min_{\theta} h(\theta) = \sum_{i=1}^{N} \frac{n_i}{\sum_{j=1}^{N} n_j} \mathcal{L}(\mathcal{D}_{i,tr}, \theta) \tag{1}$$

where N is the number of clients, n_i is the number of data points held by client i, and $\mathcal{L}(\mathcal{D}_{i,tr}, \theta) = \frac{1}{n_i} \sum_{j=1}^{n_i} \ell(z_{i,j}, \theta)$ is the empirical risk of client i. Federated learning can be performed in a cross-device or cross-silo manner[1], but in this work, we focus on the cross-silo setting [10] and from this point on, when we refer to "federated learning" we are specifically referring to cross-silo federated learning. Federated training is carried out in an iterative manner over the course of T rounds. In each round t, the server sends the current model parameters θ^t to all N clients, the clients perform R rounds of local training using their dataset $\mathcal{D}_{i,tr}$, and then send their updated model θ_i^{t+1} to the server. The server then aggregates the received models as:

$$\theta^{t+1} = \sum_{i=1}^{N} \frac{n_i}{\sum_{j=1}^{N} n_j} \theta_i^{t+1} \tag{2}$$

After aggregation, training continues until $t = T$ at which point we define $\hat{\theta} = \theta^T$ to be the final global model parameters.

3.2 Local Differential Privacy

Local differential privacy allows an analyst to learn population statistics without violating the privacy of individuals. More formally, ε-LDP is defined as follows:

Definition 1 (ε-LDP [12]). *A randomized mechanism $\mathcal{M}$ satisfies ε-LDP if and only if for any pair of input values r, r' in the domain of $\mathcal{M}$, and for any possible output $o \in Range(\mathcal{M})$, it holds:*

$$\mathbb{P}[\mathcal{M}(r) = o] \leq e^{\varepsilon} \cdot \mathbb{P}[\mathcal{M}(r') = o]$$

In other words, the probability of outputting o on record r is at most e^{ε} times the probability of outputting o on record r'.

Randomized Response One popular method used for LDP is randomized response (RR).[2] Let u be a private variable that can take one of C values. We can formalize RR as a $C \times C$ distortion matrix $\mathbf{P} = (p_{uv})_{C \times C}$ where $p_{uv} = \mathbb{P}[v|u] \in (0, 1)$ denotes the probability that the output of the RR process is $v \in \{1, \ldots, C\}$ when the real attribute value is $u \in \{1, \ldots, C\}$. Note that the entries of the distortion matrix are probabilities, and therefore the sum of the probabilities of each row is 1 [23]. Further, $\mathbf{P}$ can achieve both optimal utility and ε-DP by setting $p_{uv} = \frac{e^{\varepsilon}}{C-1+e^{\varepsilon}}$ if $u = v$ and $p_{uv} = \frac{1}{C-1+e^{\varepsilon}}$ otherwise [23].

[1] Cross-device: large federation size (100+ clients, often IoT devices like cell-phones) where the clients often only participate in a few rounds of training at most. Cross-silo: small federation size (2-10 clients, often companies or hospitals) where the clients all participate in every round of federated training.

[2] We discuss FLDP-FL under alternate LDP scenarios in our Github repository.

3.3 Influence Functions

In [13], the authors proposed the use of influence functions to study machine learning models through the lens of their training data. Specifically, they showed that the influence a single training point $z = (x, y) \in \mathcal{D}_{tr}$ has on the model parameters can be calculated without actually removing z from the training set and retraining the model on the resulting dataset. They instead simulate the removal of z by upweighting it by a small value $\frac{1}{n}$ (where n is the total number of training points). Specifically, they calculate the influence the training point has on the model parameters as:

$$\mathcal{I}_{rem,\hat{\theta}}(z) = -H_{\hat{\theta}}^{-1} \nabla_{\theta} \ell(z, \hat{\theta}) \tag{3}$$

where $H_{\hat{\theta}}^{-1}$ is the inverse Hessian matrix $H_{\hat{\theta}}^{-1} = (\frac{1}{n}\sum_{j=1}^{n} \nabla_{\hat{\theta}}^{2} \ell(z^j, \hat{\theta}))^{-1}$. We note that the inverse Hessian matrix can be calculated explicitly or efficiently estimated using conjugate gradient or stochastic estimation approaches [13].

Eq. 3 is obtained by performing a quadratic expansion around the optimal parameters $\hat{\theta}$ which gives an approximation of the function locally using information about the steepness (the gradient) and the curvature (the Hessian). Eq. 3, which gives the effect of training point z on the parameters $\hat{\theta}$, can be used to estimate the parameters that would be obtained if z was actually removed from the dataset and the model was retrained. More specifically,

$$\hat{\theta}_{-z} \approx \hat{\theta} + \frac{1}{n} \mathcal{I}_{rem,\hat{\theta}}(z) \tag{4}$$

The authors of [13] also considered the effect that modifying (rather than simply removing) a training point has on the model parameters. Consider a training point z and its modified value z_β:

$$\hat{\theta}_{z_\beta,-z} = \arg\min_{\theta \in \Theta} \mathcal{L}(Z, \theta) + \frac{1}{n}\ell(z_\beta, \theta) - \frac{1}{n}\ell(z, \theta) \tag{5}$$

is the ERM of $\hat{\theta}$ with z_β replacing z in training. The approximate effect of changing $z \rightarrow z_\beta$ on the model parameters can be computed as:

$$\mathcal{I}_{pert,\hat{\theta}}(z_\beta, -z) = -H_{\hat{\theta}}^{-1} \Big(\nabla_{\theta} \big(\ell(z_\beta, \hat{\theta}) - \ell(z, \hat{\theta}) \big) \Big) \tag{6}$$

We direct interested readers to [13] for the full derivation of Eq. 6 from Eq. 3. Eq. 6 can then be used to approximate the model parameters that would be obtained under training with z_β instead of z as:

$$\hat{\theta}_{z_\beta,-z} \approx \hat{\theta} + \frac{1}{n} \mathcal{I}_{pert,\hat{\theta}}(z_\beta, -z) \tag{7}$$

In [14] the authors showed that the influence scores calculated in Eq. 3 are additive. Namely, if we wanted to estimate the effect of a group of data points $G \subset \mathcal{D}_{tr}$ on the model, we can calculate:

$$\mathcal{I}_{rem,\hat{\theta}}(G) = -H_{\hat{\theta}}^{-1} \nabla_{\theta} \left(\sum_{j=1}^{|G|} \ell(z^j, \hat{\theta}) \right) \tag{8}$$

and use Eq. 4 to estimate the true parameters under the removal of group G.

4 Methodology

In this section, we formulate *flexible local differential privacy for federated learning* (FLDP-FL), a technique for local differentially private federated learning which grants clients the ability to change their privacy parameter ε post-training without requiring the entire federated training procedure to be repeated.

4.1 Problem Formulation

We define our federated setting as follows. We assume a horizontal cross-silo federated learning scenario where the federation is comprised of a small number of clients N (where client i has n_i data points) that participate every round and that the data is partitioned horizontally along the examples (i.e., all clients have the same feature and label domains, however, each client can have a different distribution of them). We additionally assume that each client individually randomizes their data under some client independent $\boldsymbol{\varepsilon_i} = \{\varepsilon_{i,f}\}_{f \in \mathcal{F}}$ to form $\tilde{\mathcal{D}}_{i,tr}$ before participating in federated training where $\mathcal{F}$ represents the attributes (e.g., features and/or label) in $\mathcal{D}_{i,tr}$ to be protected via randomized response. Let $f \in \mathcal{F}$ have d_f possible values. Further, let $\boldsymbol{\varepsilon_i} = \infty = \{\varepsilon_{i,f} = \infty\}_{f \in \mathcal{F}}$ represent the case where $\mathcal{D}_{i,tr} = \tilde{\mathcal{D}}_{i,tr}$ (i.e., no randomization occurs). We also assume that while only $\tilde{\mathcal{D}}_{i,tr}$ is used during federated training, clients will have access to both their un-randomized training set $\mathcal{D}_{i,tr}$ and the dataset randomized by $\boldsymbol{\varepsilon_i}$ ($\tilde{\mathcal{D}}_{i,tr}$) after training concludes. All N clients work collaboratively together to train a final global model with parameters $\hat{\theta}$. After training, client i decides to change $\boldsymbol{\varepsilon_i}$ to some $\boldsymbol{\varepsilon'_i} = \{\varepsilon'_{i,f}\}_{f \in \mathcal{F}}$ such that $\sum_{f \in \mathcal{F}} \varepsilon'_{i,f} < \sum_{f \in \mathcal{F}} \varepsilon_{i,f}$.

4.2 Perturbation influences are additive

We start by noting that while in [14] the authors showed that the influence scores calculated to simulate the *removal* of a point z are additive (see Eq. 8), it is left to be shown that influence scores that simulate the *perturbation* of a point z are additive. Here, we provide Lemma 1 which does so and provides the basis for our formulation of FLDP-FL in Section 4.3.

Lemma 1 (Perturbation influences are additive). *For a group $G \subset \mathcal{D}_{tr}$, we calculate the influence of perturbing group G to G_β, where for each point $z \in G, z \to z_\beta \in G_\beta$, as:*

$$\mathcal{I}_{pert,\hat{\theta}}(G_\beta, -G) = -H_{\hat{\theta}}^{-1} \cdot \nabla_\theta \sum_{j=1}^{|} G| \left(\ell(z_\beta^j, \hat{\theta}) - \ell(z^j, \hat{\theta}) \right) \tag{9}$$

Proof. We rewrite Eq. 6 as:

$$\mathcal{I}_{pert,\hat{\theta}}(z_\beta, -z) = -H_{\hat{\theta}}^{-1}\Big(\nabla_\theta\big(\ell(z_\beta,\hat{\theta}) - \ell(z,\hat{\theta})\big)\Big) \tag{10}$$

$$= -H_{\hat{\theta}}^{-1}\nabla_\theta\ell(z_\beta,\hat{\theta}) + H_{\hat{\theta}}^{-1}\nabla_\theta\ell(z,\hat{\theta}) \tag{11}$$

Using Eq. 8 and letting $z \in G \subset \mathcal{D}_{tr}$, $z \to z_\beta \in G_\beta$, we rewrite Eq. 11 as:

$$\mathcal{I}_{pert,\hat{\theta}}(G_\beta, -G) = -H_{\hat{\theta}}^{-1}\cdot\nabla_\theta\left(\sum_{j=1}^{|G_\beta|}\ell(z_\beta^j,\hat{\theta})\right) + H_{\hat{\theta}}^{-1}\cdot\nabla_\theta\left(\sum_{j=1}^{|G|}\ell(z^j,\hat{\theta})\right) \tag{12}$$

$$= -H_{\hat{\theta}}^{-1}\cdot\nabla_\theta\sum_{j=1}^{|G|}\Big(\ell(z_\beta^j,\hat{\theta}) - \ell(z^j,\hat{\theta})\Big) \tag{13}$$

since $|G| = |G_\beta|$ and where the last line yields Eq. 9. □

4.3 FLDP-FL

The influence functions listed so far, including Eq. 9 in Lemma 1, have been constructed to work in the traditional machine learning setting – meaning that it is assumed that the training data is located in a centralized location and can be freely accessed. Therefore, it may seem that these equations cannot be applied directly in the federated setting. However, by recognizing that each client's local dataset $\tilde{\mathcal{D}}_{i,tr}$ can be considered as a subset (i.e., *group*) from the overall federated training dataset $\boldsymbol{\mathcal{D}_{tr}} = \{\tilde{\mathcal{D}}_{i,tr}\}_{i=1}^N$ we can leverage Lemma 1 to allow client i to calculate how changing $\boldsymbol{\varepsilon_i} \to \boldsymbol{\varepsilon_i'}$ would affect $\hat{\theta}$ without requiring access to any other client's private data or having to send their private data to the global server. We now derive FLDP-FL as Theorem 1.

Theorem 1 (FLDP-FL). *Given a trained federated model $\hat{\theta}$, we can estimate the influence of client i changing $\boldsymbol{\varepsilon_i} \to \boldsymbol{\varepsilon_i'}$ on the parameters $\hat{\theta}$ as:*

$$\mathcal{I}_{pert,\hat{\theta}}^{RR}(\mathcal{D}_{i,tr}, \tilde{\mathcal{D}}_{i,tr}, \boldsymbol{\varepsilon_i'}) = -H_{\hat{\theta}}^{-1}\cdot\nabla_\theta\sum_{\substack{z\in\mathcal{D}_{i,tr},\\ \tilde{z}\in\tilde{\mathcal{D}}_{i,tr}}}\left[\left(\sum_{f_\times\in\mathcal{F}_\times} p_{f_\times}\ell(z_{f_\times},\hat{\theta})\right) - \ell(\tilde{z};\hat{\theta}))\right] \tag{14}$$

where

$$p_{f_\times} = \prod_{f\in f_\times}\mathbb{1}_{f=f_0}\left[\frac{e^{\varepsilon'_{i,f}}}{d_f - 1 + e^{\varepsilon'_{i,f}}}\right] + \mathbb{1}_{f\neq f_0}\left[\frac{1}{d_f - 1 + e^{\varepsilon'_{i,f}}}\right] \tag{15}$$

Proof. Starting from Lemma 1, let $G = \tilde{\mathcal{D}}_{i,tr}$ and $G_\beta = \mathcal{D}_{i,tr}$:

$$\mathcal{I}_{pert,\hat{\theta}}(\mathcal{D}_{i,tr}, -\tilde{\mathcal{D}}_{i,tr}) = -H_{\hat{\theta}}^{-1}\cdot\nabla_\theta\sum_{\substack{z\in\mathcal{D}_{i,tr},\\ \tilde{z}\in\tilde{\mathcal{D}}_{i,tr}}}\Big(\ell(z,\hat{\theta}) - \ell(\tilde{z},\hat{\theta})\Big) \tag{16}$$

Here, Eq. 16 gives the influence of replacing $\tilde{\mathcal{D}}_{i,tr}$ with $\mathcal{D}_{i,tr}$ during training of $\hat{\theta}$. However, we are interested in replacing $\tilde{\mathcal{D}}_{i,tr}$, which was randomized under $\boldsymbol{\varepsilon_i}$, with $\tilde{\mathcal{D}}'_{i,tr}$ that was randomized under $\boldsymbol{\varepsilon'_i}$. Let the expected value of $\ell(z, \hat{\theta})$ where $z \in \mathcal{D}_{i,tr}$ is randomized under $\boldsymbol{\varepsilon'_i}$ be written as $\mathbb{E}[\ell(z_{\boldsymbol{\varepsilon'_i}}, \hat{\theta})]$. Then:

$$\mathcal{I}_{pert,\hat{\theta}}(\mathcal{D}_{i,tr}, -\tilde{\mathcal{D}}_{i,tr}, \boldsymbol{\varepsilon'_i}) = -H_{\hat{\theta}}^{-1} \cdot \nabla_\theta \sum_{\substack{z \in \mathcal{D}_{i,tr}, \\ \tilde{z} \in \tilde{\mathcal{D}}_{i,tr}}} \Big(\mathbb{E}[\ell(z_{\boldsymbol{\varepsilon'_i}}, \hat{\theta})] - \ell(\tilde{z}, \hat{\theta}) \Big) \tag{17}$$

To find $\mathbb{E}[\ell(z_{\boldsymbol{\varepsilon'_i}}, \hat{\theta})]$, let $\mathcal{F}$ represent the set of attributes in $\mathcal{D}_{i,tr}$ to privatize under $\boldsymbol{\varepsilon'_i}$ randomized response, where each $f \in \mathcal{F}$ has d_f possible values as well as an independent privacy parameter $\varepsilon'_{i,f}$. Further, let $\mathcal{F}_\times$ represent the cartesian product[3] of $f \in \mathcal{F}$ and f_0 represent the original attribute value of f in data point $z \in \mathcal{D}_{i,tr}$. Under $\boldsymbol{\varepsilon'_i}$ randomized response, $z \in \mathcal{D}_{i,tr}$ can take one of $2^{|\mathcal{F}|}$ values $f_\times \in \mathcal{F}_\times$, each with probability $p_{f_\times} = \prod_{f \in f_\times} \mathbb{1}_{f=f_0} \left[\frac{e^{\varepsilon'_{i,f}}}{d_f - 1 + e^{\varepsilon'_{i,f}}} \right] + \mathbb{1}_{f \neq f_0} \left[\frac{1}{d_f - 1 + e^{\varepsilon'_{i,f}}} \right]$ (see Section 3.2). The expected value of the loss where $z \in \mathcal{D}_{i,tr}$ is randomized under $\boldsymbol{\varepsilon'_i}$ can therefore be written as:

$$\mathbb{E}[\ell(z_{\boldsymbol{\varepsilon'_i}}, \hat{\theta})] = \sum_{f_\times \in \mathcal{F}_\times} p_{f_\times} \ell(z_{f_\times}, \hat{\theta}) \tag{18}$$

where $\tilde{z}$ and $z_{f_\times}$ are *equivalent* minus the attributes in $\mathcal{F}_\times$ which have been replaced according to $f_\times$. Substituting the right hand side of Eq. 18 into Eq. 17 yields Eq. 14. $\square$

To obtain the estimated global model parameters where a client re-randomizes their data under $\boldsymbol{\varepsilon_i} \to \boldsymbol{\varepsilon'_i}$ and federated training is re-performed, we calculate:

$$\hat{\theta} = \hat{\theta} + \frac{1}{\sum_{j=1}^{N} n_j} \mathcal{I}^{\text{RR}}_{pert,\hat{\theta}}(\mathcal{D}_{i,tr}, \tilde{\mathcal{D}}_{i,tr}, \boldsymbol{\varepsilon'_i}) \tag{19}$$

where $\sum_{j=1}^{N} n_j$ is the sum of the data used in the federated learning process by all N clients. In Section 5 we analyze the setting where $m > 1$ clients simultaneously update $\boldsymbol{\varepsilon_i} \to \boldsymbol{\varepsilon'_i}$ (which could arise in instances where wide sweeping government regulation is updated). Here, we leverage Lemma 1 which says that perturbation influences are additive, and calculate the estimated model parameters under the m clients updating $\boldsymbol{\varepsilon_i} \to \boldsymbol{\varepsilon'_i}$ as:

$$\hat{\theta} = \hat{\theta} + \frac{1}{\sum_{j=1}^{N} n_j} \sum_{i=1}^{m} \mathcal{I}_i \tag{20}$$

where each client i calculates $\mathcal{I}_i$ using Eq. 14 independently.

[3] For example, if $\mathcal{F} = \{\text{gender}, \text{income}\}$ where gender$= \{m, f\}$ and income$= \{0, 1\}$ then $\mathcal{F}_\times = \{(m, 0), (m, 1), (f, 0), (f, 1)\}$.

Here, we show the derivation for Eq. 20 which enables multiple clients to update $\boldsymbol{\varepsilon_i} \to \boldsymbol{\varepsilon'_i}$ simultaneously. Let $G_i = \tilde{\mathcal{D}}_{i,tr}$ and $G_{i,\beta} = \mathcal{D}_{i,tr}$ for all m clients i who desire to change $\boldsymbol{\varepsilon_i} \to \boldsymbol{\varepsilon'_i}$. Since each G_i can be seen as an independent subset of the overall federated training set $\boldsymbol{\mathcal{D}_{tr}} = \tilde{\mathcal{D}}_{1,tr} \cup \cdots \cup \tilde{\mathcal{D}}_{N,tr}$, we can say that the union of G_i is also a subset of $\boldsymbol{\mathcal{D}_{tr}}$. I.e., $\boldsymbol{G} = G_1 \cup \ldots G_m \subset \boldsymbol{\mathcal{D}_{tr}}$. Therefore, we can follow a derivation similar to Theorem 1 to form the optimization function to support multiple clients updating their privacy parameters simultaneously.

Let $\boldsymbol{G_\beta} = G_{1,\beta} \cup \cdots \cup G_{m,\beta}$ and similarly let $\boldsymbol{G} = G_1 \cup \cdots \cup G_m$. Starting from Eq. 12 in Lemma 1, we can write:

$$\mathcal{I}_{pert,\hat{\theta}}(\boldsymbol{G_\beta}, -\boldsymbol{G}) = -H_{\hat{\theta}}^{-1} \nabla_\theta \sum_{z_\beta \in \boldsymbol{G_\beta}} \ell(z_\beta, \hat{\theta}) + H_{\hat{\theta}}^{-1} \nabla_\theta \sum_{z \in \boldsymbol{G}} \ell(z, \hat{\theta}) \tag{21}$$

$$= -H_{\hat{\theta}}^{-1} \nabla_\theta \sum_{i=1}^{m} \sum_{z_{i,\beta} \in G_{i,\beta}} \ell(z_{i,\beta}, \hat{\theta}) + H_{\hat{\theta}}^{-1} \nabla_\theta \sum_{i=1}^{m} \sum_{z_i \in G_i} \ell(z_i, \hat{\theta}) \tag{22}$$

$$= \sum_{i=1}^{m} -H_{\hat{\theta}}^{-1} \nabla_\theta \sum_{\substack{z_i \in G_i, \\ z_{i,\beta} \in G_{i,\beta}}} \ell(z_{i,\beta}, \hat{\theta}) - \ell(z_i, \hat{\theta}) \tag{23}$$

$$\tag{24}$$

since $|G_i| = |G_{i,\beta}|$. Then, replacing $\ell(z_{i,\beta}, \hat{\theta})$ with $\mathbb{E}[\ell(z_{\boldsymbol{\varepsilon'_i}}, \hat{\theta})]$ yields

$$\mathcal{I}_{pert,\hat{\theta}}^{\mathrm{RR}}(\boldsymbol{G_\beta}, -\boldsymbol{G}, \boldsymbol{\varepsilon'}) = \sum_{i=1}^{m} \mathcal{I}_{pert,\hat{\theta}}^{\mathrm{RR}}(G_{i,\beta}, -G_i, \boldsymbol{\varepsilon'_i}) \tag{25}$$

and $\boldsymbol{\varepsilon} = \{\boldsymbol{\varepsilon'_i}\}_{i=1}^{m}$. Then, to obtain the estimate for the parameters where all m clients simultaneously change $\boldsymbol{\varepsilon_i} \to \boldsymbol{\varepsilon'_i}$, we can calculate:

$$\hat{\theta} = \hat{\theta} + \frac{1}{\sum_{j=1}^{N} n_j} \sum_{i=1}^{m} \mathcal{I}_{pert,\hat{\theta}}^{\mathrm{RR}}(G_{i,\beta}, -G_i, \boldsymbol{\varepsilon'_i}) \tag{26}$$

which recovers Eq. 20.

We provide an overview of our FLDP-FL process (specifically for one client updating $\boldsymbol{\varepsilon_i} \to \boldsymbol{\varepsilon'_i}$) in Algorithm 1. Here, the client generates $\mathcal{I}$ according to Eq. 14 to calculate the influence on the global model of locally updating $\boldsymbol{\varepsilon_i} \to \boldsymbol{\varepsilon'_i}$ and sends it to the server. The server then updates the model parameters $\hat{\theta}$ according to Eq. 19 and distributes it to the clients.

Two natural questions to the derivation of Theorem 1 are: 1) why we use a weighted variant of the expected loss (Eq. 18) rather than a simple average; and 2) why Eq. 15 produces the correct weights for the weighted average. To answer question one, since we do not know a priori which combination $f_\times \in \mathcal{F}_\times$ will be produced by the randomized response process, we need to consider that the replacement of $\tilde{z}$ could be done by any combination $f_\times \in \mathcal{F}_\times$. However, the probability distribution over $\mathcal{F}_\times$ is not uniform and therefore taking a simple

Algorithm 1 FLDP-FL

```
1: Each client randomizes $\mathcal{D}_{i,tr}$ under $\boldsymbol{\varepsilon_i}$ randomized response to produce $\tilde{\mathcal{D}}_{i,tr}$
2: Traditional FedAvg [16] training is carried out where each client participates with
   $\tilde{\mathcal{D}}_{i,tr}$ to produce final global model parameters $\hat{\theta}$
3: if Client $i$ updates $\boldsymbol{\varepsilon_i} \to \boldsymbol{\varepsilon'_i}$ then
4:     Client $i$ computes:
       $$\mathcal{I} \leftarrow -H_{\hat{\theta}}^{-1} \nabla_\theta \sum_{\substack{z \in \mathcal{D}_{i,tr}, \\ \tilde{z} \in \tilde{\mathcal{D}}_{i,tr}}} \left[ \left( \sum_{f_\times \in \mathcal{F}_\times} p_{f_\times} \ell(z_{f_\times}, \hat{\theta}) \right) - \ell(\tilde{z}; \hat{\theta})) \right]$$
       and sends to server
5:     Server computes:
       $$\hat{\theta} = \hat{\theta} + \frac{1}{\sum_{j=1}^{N} n_j} \mathcal{I}$$
       and sends to all clients
```

average of the losses produced by $\{z_{f_\times}\}_{f_\times \in \mathcal{F}_\times}$ will not produce the true expected loss. Therefore, to generate the expected loss, we need to multiply $\ell(z_{f_\times}, \hat{\theta})$ by its probability $p_{f_\times}$ generated by Eq. 15. This leads to the second question, which is why the formulation of Eq. 15 produces the correct weights. Since we assume each attribute in $\mathcal{F}$ to be independent, we multiply the probability of each attribute $f \in \mathcal{F}$ by $\frac{e^{\varepsilon'_{i,f}}}{d_f - 1 + e^{\varepsilon'_{i,f}}}$ if f is equal to the original attribute value f_0 in $z \in \mathcal{D}_{i,tr}$ (*not* $\tilde{z} \in \tilde{\mathcal{D}}_{i,tr}$ since we want to randomize the original data, not the data perturbed by $\boldsymbol{\varepsilon_i}$) or by $\frac{1}{d_f - 1 + e^{\varepsilon'_{i,f}}}$ if $f \neq f_0$ (see Section 3.2).

4.4 Discussion

Computation and Communication Cost: In [18], the authors note that using influence functions in the federated learning setting faces many challenges. First, the authors point out that even if implicit Hessian-vector products are used to overcome the cost of forming and inverting the Hessian of the empirical risk (which costs $O(nm^2 + m^3)$ where n is the number of training points and m is the number of parameters), it is still communication intensive if the influence is calculated every round as it requires the transfer of all training data to the global server. In our work, however, not only does the the influence function have to be calculated *sparingly* (i.e., only when the client updates their privacy level $\boldsymbol{\varepsilon_i} \to \boldsymbol{\varepsilon'_i}$), ***we require the client who desires to update*** $\boldsymbol{\varepsilon_i} \to \boldsymbol{\varepsilon'_i}$ ***to calculate Eq. 14***. In the federated setting, the global model does not have access to the client's data, which is required when calculating Eq. 14. However, the client naturally has access to the final global model parameters $\hat{\theta}$ and therefore has all the required information to calculate Eq. 14. Additionally, this puts the burden of calculating the influence scores on the client doing the change, not the global model or the other clients who simply have to update the model parameters.

Table 1. Time complexity of FLDP-FL versus full federated retraining. X: exact, CG: conjugate gradient, SE: stochastic estimation, n: total number of training points, n_i: number of training points of client i, p: number of parameters, r: recursion depth for SE estimation, t: number of recursions for SE estimation, E: number of federated epochs.

FLDP-FL (X)	FLDP-FL (CG)	FLDP-FL (SE)	Fed. Retrain
$\mathcal{O}(n_i p^2 + p^3)$	$\mathcal{O}(n_i p)$	$\mathcal{O}(n_i p + rtp)$	$\mathcal{O}(Enp)$

We also clarify that not only does Eq. 14 have to be calculated sparingly, the computational cost to calculate Eq. 14 is much lower than that of total federated model retraining, especially if approaches such as stochastic estimation or conjugate gradient are used to approximate the Hessian. We point interested readers to [3,13] for a more in depth discussion of how these approximations can decrease the total computation time. We additionally provide concrete experimental results in Section 5 that support this claim and here briefly expound upon the computational cost of FLDP-FL versus full retraining.

In Table 1, we detail the computational complexity of calculating the influence function using three different approaches to computing the inverse Hessian as well as the time complexity of normal training of a logistic regression model using gradient based learning. Specifically, we detail the complexity of computing the Hessian for Eq. 14 explicitly and using the conjugate gradient (CG) or stochastic estimation (SE) approaches to estimate it. While using the explicit Hessian approach seems to be more computationally complex than retraining, we note that only the client changing $\boldsymbol{\varepsilon_i} \rightarrow \boldsymbol{\varepsilon'_i}$ has to calculate the Hessian whereas all clients have to participate in retraining.

One of the main bottlenecks to federated learning is the communication cost of the clients sending/receiving parameter updates from the server as it is common that communication will be limited by an upload bandwidth of 1 MB/s or less [16]. These costs are influence by various parameters such as the client's dataset size, the size of the model, and the number of clients participating. Computing Eq. 14 takes minimal communication (i.e., client i ending $\mathcal{I}$ to the server and the server sending the updated $\hat{\theta}$ to all clients), whereas in full federated retraining, each client must participate in all E rounds, and perform both a download from the global server at the start of the round and send the updated parameters back to the global server at the end of the round. I.e., FLDP-FL takes a maximum of $N + 1$ communications while full federated retraining takes $2EN$.

Connection to Machine Unlearning: We also point out that FLDP-FL has an intuitive connection to machine unlearning. Under new privacy regulations such as the GDPR, consumers have to be afforded the "right to be forgotten". Therefore, in addition to our setting of a client having to update $\boldsymbol{\varepsilon_i} \rightarrow \boldsymbol{\varepsilon'_i}$, it could also be the case that a client may have to be removed from training entirely. Machine unlearning, the process of modifying a machine learning model to forget parts of data that it was trained on, has risen in popularity to satisfy these new privacy regulations. Unlearning has also been studied in the federated learning setting, albeit less extensively than in the standard machine learning setting.

Using influence functions, we can estimate the removal of a client i from training without having to redo federated training on the smaller client set. Specifically, using a formulation similar to Eq. 3, we can write:

$$\mathcal{I}_{rem,\hat{\theta}}(\tilde{\mathcal{D}}_{i,tr}) = -H_{\hat{\theta}}^{-1} \cdot \nabla_\theta \sum_{\tilde{z} \in \tilde{\mathcal{D}}_{i,tr}} \ell(\tilde{z}; \hat{\theta}) \tag{27}$$

where $\hat{\theta} = \hat{\theta} + \frac{1}{\sum_{j=1}^{N} n_j} \mathcal{I}_{rem,\hat{\theta}}(\tilde{\mathcal{D}}_{i,tr})$ gives an estimation of the true model parameters achieved when a client is removed from training entirely.

We note that in this work we do not study the entire removal of a client and further clarify that the setting where $\boldsymbol{\varepsilon}_{\boldsymbol{i}}' = 0 = \{\epsilon_{i,f}' = 0\}_{f \in \mathcal{F}}$ is *not equivalent* to machine unlearning as we are replacing client i's data with random noise (which can affect the performance of the global model) and not removing it entirely. This can be seen by comparing Eq. 14 with Eq. 27. In Eq. 27, we simply have to take the gradient of the loss of the data points used in training, which is then used to simulate the removal of these points from training. In Eq. 14 however, we include the addition of the expected loss of the training points under perturbation by $\boldsymbol{\varepsilon}_{\boldsymbol{i}}'$. Even though setting $\boldsymbol{\epsilon}_{\boldsymbol{i}}' = \infty$ would effectively make the clients data pure noise, and in a sense causes the model to unlearn the true data $\tilde{\mathcal{D}}_{i,tr}$, it will ultimately negatively affect the performance of the global model due to the introduction of random noise. On the other hand, removing a client entirely using Eq. 27, or other techniques proposed by work like [22,25] most likely will not see significant degradation in performance, especially when additional optimization is used to preserve model performance under client removal. Our work instead can be seen as an irregular instance of unlearning where instead of forgetting certain data points entirely, we instead want to change how well the model is able to understand the relationship between the data features and label (e.g., by increasing the noise applied during differential privacy).

5 Evaluation

In this section, we evaluate the ability of our formulated FLDP-FL approach for estimating the true change in the model that would occur if a client (after initial federated training) updated their privacy parameter $\boldsymbol{\varepsilon}_{\boldsymbol{i}} \to \boldsymbol{\varepsilon}_{\boldsymbol{i}}'$ and the federated learning process repeated. Here, we are interested in answering the following:

Q1. Under one client changing $\boldsymbol{\varepsilon}_{\boldsymbol{i}} \to \boldsymbol{\varepsilon}_{\boldsymbol{i}}'$, how does the model estimated in Alg. 1 compare in terms of test loss, test accuracy, and distance with the true model obtained when full federated re-training occurs?

Q2. Is our FLDP-FL approach able to properly estimate the true model when multiple clients simultaneously change their privacy value from $\boldsymbol{\varepsilon}_{\boldsymbol{i}} \to \boldsymbol{\varepsilon}_{\boldsymbol{i}}'$?

Q3. How does the size of the client performing the change from $\boldsymbol{\varepsilon}_{\boldsymbol{i}} \to \boldsymbol{\varepsilon}_{\boldsymbol{i}}'$ affect the ability of our FLDP-FL approach to estimate the true model parameters?

Q4. How efficient (in terms of computation time) is FLDP-FL compared to full federated model retraining when one client decides to update $\boldsymbol{\varepsilon_i} \rightarrow \boldsymbol{\varepsilon_i'}$?

5.1 Datasets, Federated Setting, and Baselines

In this work, we test FLDP-FL using three datasets: ACS Income [5], Glioma [20], and ACS Public Coverage [5]. We divide the data among the clients according to the process described in [16] such that the clients' local datasets are saturated by one label, and therefore, if they attempted to train a model on their own, they would obtain a model with poor generalization performance. For Q1/Q2/Q4 all clients are allotted the same number of data points, while in Q3 we allocate more data points to the client changing $\boldsymbol{\varepsilon_i} \rightarrow \boldsymbol{\varepsilon_i'}$. We allow the global model to be made of a single fully-connected layer, use SGD for optimization, and use cross-entropy for loss. Further, we set the number of clients to 5, assume all clients have access to $\mathcal{D}_{i,tr}$ (their un-randomized training set) and $\tilde{\mathcal{D}}_{i,tr}$ (which is formed by randomizing $\mathcal{D}_{i,tr}$ by $\boldsymbol{\varepsilon_i}$), and each participant participates in every round of model training. Since we are (to our knowledge) the first work to consider how a client can change their $\boldsymbol{\varepsilon_i}$ after training, we only compare against is naïve retraining, where for each setting, we re-randomize the changing client's data under $\boldsymbol{\varepsilon_i'}$ and re-preform the federated training procedure. We test a wide range of $\boldsymbol{\varepsilon_i} = x = \{\varepsilon_{i,f} = x\}_{f \in \mathcal{F}}$ values $\{\infty, 5, 4, 3, 2, 1\}$ as well as a wide range of $\boldsymbol{\varepsilon_i'} = x = \{\varepsilon_{i,f}' = x\}_{f \in \mathcal{F}}$ values $\{5, 4, 3, 2, 1, 0.5, 0.1\}$ in the analysis of Q1-Q4. For simplicity, we have all clients use the same $\boldsymbol{\varepsilon_i}/\boldsymbol{\varepsilon_i'}$. We run all experiments three times and report the average. Further dataset and experimental details such as which attributes $\mathcal{F}$ were protected under randomized response and the selected hyperparameters can be found in our Github repository `https://shorturl.at/qFRJD`.

5.2 Evaluation Metrics

We consider three primary metrics to evaluate our proposed FLDP-FL method: 1) average loss difference (ALD), average accuracy difference (AAD), and euclidean distance (ED).

$$ALD = \mathcal{L}(\mathcal{D}_{te}, \theta_{\mathcal{I}}) - \mathcal{L}(\mathcal{D}_{te}, \theta_{\boldsymbol{\varepsilon_i} \rightarrow \boldsymbol{\varepsilon_i'}}) \tag{28}$$

$$AAD = \mathrm{Acc}(\mathcal{D}_{te}, \theta_{\mathcal{I}}) - \mathrm{Acc}(\mathcal{D}_{te}, \theta_{\boldsymbol{\varepsilon_i} \rightarrow \boldsymbol{\varepsilon_i'}}) \tag{29}$$

$$ED = ||\theta_{\mathcal{I}} - \theta_{\boldsymbol{\varepsilon_i} \rightarrow \boldsymbol{\varepsilon_i'}}||_2 \tag{30}$$

where $\mathcal{L}(\mathcal{D}_{te}, \cdot)$ denotes the test loss, $\mathrm{Acc}(\mathcal{D}_{te}, \cdot)$ denotes the test accuracy, $\theta_{\mathcal{I}}$ represents the influence-estimated parameters using Alg. 1, and $\theta_{\boldsymbol{\varepsilon_i} \rightarrow \boldsymbol{\varepsilon_i'}}$ represents the true parameters obtained when client i's data is re-randomized under $\boldsymbol{\varepsilon_i'}$ and federated retraining is re-performed. For all ALD, AAD, and ED, values closer to zero are desirable. We note that these values are only used for our analysis of FLDP-FL's estimation ability and that in practice, they may not need to be computed. However, when necessary, the global server is the one who would calculate $\mathcal{L}(\mathcal{D}_{te}, \theta_{\mathcal{I}})$ and $\mathrm{Acc}(\mathcal{D}_{te}, \theta_{\mathcal{I}})$ as the clients do not have access to $\mathcal{D}_{te}$.

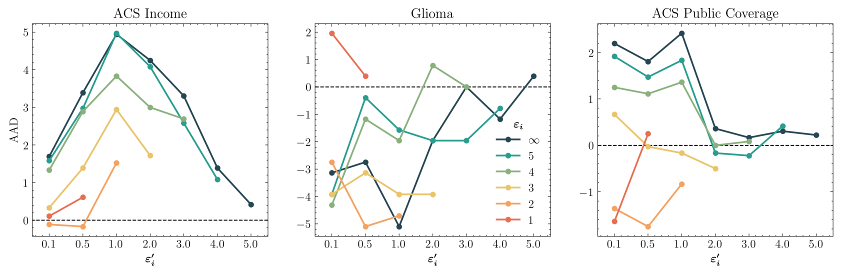

Fig. 1. Average accuracy difference (AAD, Eq. 29) when one client changes $\varepsilon_i \rightarrow \varepsilon'_i$.

5.3 Analysis

Q1. We begin our evaluation by studying the ability of FLDP-FL to estimate the model parameters that result from a single client changing $\varepsilon_i \rightarrow \varepsilon'_i$ after federated training concludes. After training the global model using standard FedAvg, we randomly select one client to calculate Eq. 14. The server then calculates the resulting test loss and accuracy under the newly estimated model parameters (derived using Eq. 19). We plot the results for AAD in Fig. 1 and additionally list the results for all metrics (along with their standard deviations) in Table 2. Across all three datasets, ALD trends closer to zero as ε'_i approaches ε_i, however, even in cases when the difference between the two is large (e.g., $\varepsilon_i = \infty$ and $\varepsilon'_i = 0.1$), the difference in the estimated and true loss is small. Similar trends hold for both AAD and ED, although the trend for AAD is slightly weaker (e.g., ACS Income obtains lower AAD for $\varepsilon'_i = 0.1/0.5$ than ε'_i=1). Again, however, even when the difference is large, the AAD value is still within $\pm 5\%$ of the true accuracy. In general, the results for the Glioma dataset are worse than that of ACS Income or ACS Public Coverage and we attribute this to the dataset being small ($\sim$400 data points). In future work, we plan to more rigorously analyze how overall federated dataset size affects the ability of FLDP-FL to generate proper estimations for the model parameters. Overall, these results support that FLDP-FL is able to estimate the model parameters resulting from a single client changing $\varepsilon_i \rightarrow \varepsilon'_i$ after federated training.

Q2. To analyze the ability of FLDP-FL in estimating the true model under multiple clients changing $\varepsilon_i \rightarrow \varepsilon'_i$, we again follow a setup where first we train the federated model as normal with each client randomizing their data using ε_i randomized response before participating in training. We randomly select $m = \{1, 2, 3\}$ clients to change $\varepsilon_i \rightarrow \varepsilon'_i$ and calculate Eq. 14 to obtain $\mathcal{I}_i$. We then estimate $\hat{\theta}$ using Eq. 20. We plot the results for all datasets where $\varepsilon_i = 3 \rightarrow \varepsilon'_i = 1$ in the top row of Fig. 2. For the two larger datasets, ACS Income and Public Coverage, the ability of FLDP-FL to estimate the true parameters maintains consistent as the number of clients changing $\varepsilon_i = 3 \rightarrow \varepsilon'_i = 1$ increases. On the other hand, when increasing the number of changing clients on small datasets, the quality of the estimation provided by FLDP-FL degrades quite significantly.

Table 2. Results for average loss difference (ALD, Eq. 28), average accuracy difference (AAD, Eq. 29), and euclidean distance (ED, Eq. 30) for parameters estimated by FLDP-FL when one client changes $\boldsymbol{\varepsilon_i} \rightarrow \boldsymbol{\varepsilon'_i}$.

ε_i	ε'_i	ACS Income			Glioma			ACS Public Coverage		
		ALD	AAD	ED	ALD	AAD	ED	ALD	AAD	ED
	0.1	$-0.026_{\pm 0.005}$	$1.69_{\pm 0.10}$	$0.136_{\pm 0.040}$	$0.096_{\pm 0.031}$	$-3.14_{\pm 1.47}$	$1.462_{\pm 0.283}$	$-0.014_{\pm 0.005}$	$2.19_{\pm 1.52}$	$0.192_{\pm 0.074}$
	0.5	$-0.029_{\pm 0.003}$	$3.39_{\pm 0.31}$	$0.111_{\pm 0.027}$	$0.066_{\pm 0.021}$	$-2.75_{\pm 3.09}$	$1.321_{\pm 0.287}$	$-0.014_{\pm 0.003}$	$1.81_{\pm 0.21}$	$0.178_{\pm 0.010}$
	1	$-0.043_{\pm 0.007}$	$4.94_{\pm 0.70}$	$0.132_{\pm 0.008}$	$0.044_{\pm 0.023}$	$-5.10_{\pm 3.64}$	$1.211_{\pm 0.163}$	$-0.007_{\pm 0.001}$	$2.42_{\pm 1.21}$	$0.159_{\pm 0.039}$
∞	2	$-0.022_{\pm 0.003}$	$4.25_{\pm 0.49}$	$0.093_{\pm 0.009}$	$0.038_{\pm 0.040}$	$-1.96_{\pm 2.22}$	$0.902_{\pm 0.495}$	$0.005_{\pm 0.001}$	$0.36_{\pm 0.62}$	$0.106_{\pm 0.041}$
	3	$-0.012_{\pm 0.001}$	$3.31_{\pm 0.71}$	$0.044_{\pm 0.008}$	$0.009_{\pm 0.009}$	$0.00_{\pm 0.96}$	$0.471_{\pm 0.304}$	$0.005_{\pm 0.003}$	$0.17_{\pm 0.14}$	$0.138_{\pm 0.057}$
	4	$-0.004_{\pm 0.002}$	$1.39_{\pm 0.75}$	$0.023_{\pm 0.003}$	$0.006_{\pm 0.010}$	$-1.18_{\pm 1.66}$	$0.271_{\pm 0.219}$	$0.001_{\pm 0.001}$	$0.31_{\pm 0.22}$	$0.057_{\pm 0.029}$
	5	$-0.001_{\pm 0.001}$	$0.42_{\pm 0.25}$	$0.012_{\pm 0.002}$	$0.001_{\pm 0.002}$	$0.39_{\pm 0.55}$	$0.155_{\pm 0.075}$	$-0.001_{\pm 0.00}$	$0.22_{\pm 0.14}$	$0.020_{\pm 0.008}$
	0.1	$-0.017_{\pm 0.001}$	$1.58_{\pm 0.12}$	$0.131_{\pm 0.038}$	$0.087_{\pm 0.038}$	$-3.92_{\pm 2.22}$	$1.600_{\pm 0.287}$	$-0.011_{\pm 0.004}$	$1.92_{\pm 1.73}$	$0.182_{\pm 0.064}$
	0.5	$-0.025_{\pm 0.002}$	$2.97_{\pm 0.31}$	$0.108_{\pm 0.026}$	$0.057_{\pm 0.025}$	$-0.39_{\pm 1.11}$	$1.117_{\pm 0.279}$	$-0.010_{\pm 0.003}$	$1.47_{\pm 0.37}$	$0.165_{\pm 0.005}$
5	1	$-0.038_{\pm 0.005}$	$4.97_{\pm 0.21}$	$0.126_{\pm 0.010}$	$0.035_{\pm 0.021}$	$-1.57_{\pm 3.09}$	$1.107_{\pm 0.006}$	$-0.004_{\pm 0.001}$	$1.83_{\pm 0.72}$	$0.178_{\pm 0.022}$
	2	$-0.022_{\pm 0.005}$	$4.08_{\pm 0.30}$	$0.083_{\pm 0.016}$	$0.021_{\pm 0.026}$	$-1.96_{\pm 2.93}$	$0.800_{\pm 0.264}$	$0.005_{\pm 0.001}$	$-0.17_{\pm 0.34}$	$0.145_{\pm 0.043}$
	3	$-0.010_{\pm 0.001}$	$2.58_{\pm 0.62}$	$0.044_{\pm 0.011}$	$0.002_{\pm 0.009}$	$-1.96_{\pm 1.47}$	$0.457_{\pm 0.156}$	$0.007_{\pm 0.003}$	$-0.22_{\pm 0.04}$	$0.149_{\pm 0.043}$
	4	$-0.002_{\pm 0.001}$	$1.08_{\pm 0.68}$	$0.025_{\pm 0.007}$	$-0.004_{\pm 0.006}$	$-0.78_{\pm 0.55}$	$0.265_{\pm 0.197}$	$0.002_{\pm 0.002}$	$0.42_{\pm 0.34}$	$0.057_{\pm 0.029}$
	0.1	$-0.006_{\pm 0.005}$	$1.33_{\pm 0.07}$	$0.094_{\pm 0.048}$	$0.075_{\pm 0.035}$	$-4.31_{\pm 1.47}$	$1.768_{\pm 0.046}$	$-0.007_{\pm 0.004}$	$1.25_{\pm 1.54}$	$0.188_{\pm 0.045}$
	0.5	$-0.012_{\pm 0.010}$	$2.89_{\pm 0.41}$	$0.081_{\pm 0.026}$	$0.053_{\pm 0.035}$	$-1.18_{\pm 1.92}$	$1.197_{\pm 0.245}$	$-0.008_{\pm 0.002}$	$1.11_{\pm 0.39}$	$0.157_{\pm 0.014}$
4	1	$-0.029_{\pm 0.004}$	$3.83_{\pm 0.20}$	$0.119_{\pm 0.005}$	$0.028_{\pm 0.022}$	$-1.96_{\pm 2.42}$	$1.055_{\pm 0.126}$	$-0.002_{\pm 0.001}$	$1.36_{\pm 0.63}$	$0.152_{\pm 0.033}$
	2	$-0.012_{\pm 0.003}$	$3.00_{\pm 0.34}$	$0.086_{\pm 0.005}$	$0.009_{\pm 0.020}$	$0.78_{\pm 0.55}$	$0.723_{\pm 0.176}$	$0.005_{\pm 0.001}$	$0.00_{\pm 0.61}$	$0.108_{\pm 0.061}$
	3	$-0.008_{\pm 0.001}$	$2.69_{\pm 0.55}$	$0.044_{\pm 0.011}$	$-0.002_{\pm 0.012}$	$-0.00_{\pm 0.96}$	$0.442_{\pm 0.223}$	$0.001_{\pm 0.001}$	$0.08_{\pm 0.31}$	$0.070_{\pm 0.032}$
	0.1	$0.006_{\pm 0.007}$	$0.33_{\pm 0.20}$	$0.107_{\pm 0.022}$	$0.060_{\pm 0.040}$	$-3.92_{\pm 2.93}$	$1.470_{\pm 0.328}$	$0.008_{\pm 0.012}$	$0.67_{\pm 1.59}$	$0.118_{\pm 0.027}$
3	0.5	$0.005_{\pm 0.004}$	$1.39_{\pm 0.08}$	$0.085_{\pm 0.021}$	$0.039_{\pm 0.032}$	$-3.14_{\pm 1.47}$	$1.167_{\pm 0.377}$	$0.005_{\pm 0.006}$	$-0.03_{\pm 0.63}$	$0.105_{\pm 0.029}$
	1	$-0.014_{\pm 0.006}$	$2.94_{\pm 0.61}$	$0.101_{\pm 0.005}$	$0.039_{\pm 0.032}$	$-3.92_{\pm 2.00}$	$1.230_{\pm 0.236}$	$0.004_{\pm 0.004}$	$-0.17_{\pm 1.18}$	$0.162_{\pm 0.042}$
	2	$-0.007_{\pm 0.003}$	$1.72_{\pm 0.91}$	$0.053_{\pm 0.010}$	$0.028_{\pm 0.033}$	$-3.92_{\pm 2.22}$	$1.079_{\pm 0.312}$	$0.010_{\pm 0.009}$	$-0.50_{\pm 0.72}$	$0.123_{\pm 0.071}$
	0.1	$0.017_{\pm 0.004}$	$-0.11_{\pm 0.34}$	$0.072_{\pm 0.013}$	$0.031_{\pm 0.011}$	$-2.75_{\pm 1.47}$	$1.295_{\pm 0.197}$	$0.014_{\pm 0.010}$	$-1.36_{\pm 0.98}$	$0.179_{\pm 0.085}$
2	0.5	$0.008_{\pm 0.004}$	$-0.17_{\pm 0.54}$	$0.061_{\pm 0.004}$	$0.034_{\pm 0.014}$	$-5.10_{\pm 0.55}$	$1.018_{\pm 0.367}$	$0.012_{\pm 0.005}$	$-1.75_{\pm 0.34}$	$0.252_{\pm 0.055}$
	1	$-0.003_{\pm 0.003}$	$1.53_{\pm 1.58}$	$0.064_{\pm 0.008}$	$0.027_{\pm 0.002}$	$-4.71_{\pm 3.46}$	$1.277_{\pm 0.163}$	$0.005_{\pm 0.001}$	$-0.83_{\pm 0.72}$	$0.184_{\pm 0.056}$
1	0.1	$0.010_{\pm 0.002}$	$0.11_{\pm 0.27}$	$0.054_{\pm 0.020}$	$-0.009_{\pm 0.023}$	$1.96_{\pm 2.22}$	$1.510_{\pm 0.427}$	$0.004_{\pm 0.005}$	$-0.78_{\pm 0.22}$	$0.185_{\pm 0.098}$
	0.5	$0.002_{\pm 0.002}$	$0.61_{\pm 0.45}$	$0.049_{\pm 0.006}$	$-0.005_{\pm 0.022}$	$0.39_{\pm 3.88}$	$1.394_{\pm 0.358}$	$0.008_{\pm 0.007}$	$-0.58_{\pm 0.27}$	$0.172_{\pm 0.048}$

Specifically, for Glioma, FLDP-FL's estimated parameters consistently underperforms the performance obtained under the true retrained parameters (e.g., AAD of -17.5% when 3 clients are changed). These results show that FLDP-FL is able to estimate the true parameters under multiple clients changing $\boldsymbol{\varepsilon_i} \rightarrow \boldsymbol{\varepsilon'_i}$ when the dataset is not excessively small. When the number of clients changing becomes too high, we recommend retraining the federated model instead of using estimations provided by FLDP-FL.

Q3. To analyze the ability of FLDP-FL to estimate the true model parameters when the client changing $\boldsymbol{\varepsilon_i} \rightarrow \boldsymbol{\varepsilon'_i}$ makes up a large portion of the overall federated dataset, we still perform federated training as normal. However, instead of all clients having a relatively equal amount of data points, we allocate the changing client X% of the overall federated dataset and distribute the remaining 1-X% to the other four clients. We show the results when $\boldsymbol{\varepsilon_i} = 3 \rightarrow \boldsymbol{\varepsilon'_i} = 1$ in the bottom row of Fig. 2. There is an obvious trend of the estimation provided by FLDP-FL becoming worse as the dataset percentage makeup of the changing

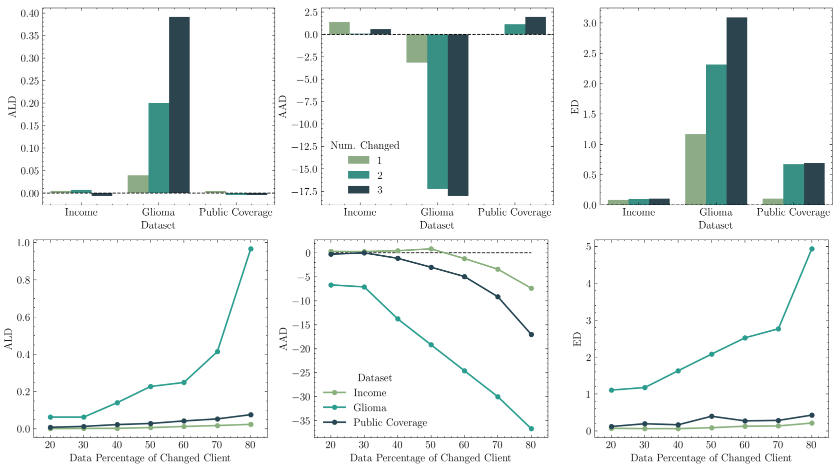

Fig. 2. Average loss difference (ALD, Eq. 28), average accuracy difference (AAD, Eq. 29), and Euclidean distance (ED, Eq. 30) when $\varepsilon_i = 3 \rightarrow \varepsilon'_i = 1$ for: *Top*: multiple clients changing; and *Bottom*: different changing client dataset size.

client increases. This is not surprising as there is a degree of randomness to the FLDP-FL estimation, and increasing the size of the dataset changing increases the amount of noise in the estimation. However, ALD degrades gracefully, especially in cases where the dataset is large (i.e., ACS Income and ACS Public Coverage). AAD suffers however when more than 50% of the data belongs to the changing client. In cases where the client changing owns the majority of the federated data, it may be advantageous to perform federated retraining over estimation via FLDP-FL.

Q4. In Table 3 we report the average time to train one full federated learning model and the time to compute Eq. 14 using an explicit Hessian calculation (e.g., no estimation approaches that would further reduce the computational time were used) under one client changing $\varepsilon_i \rightarrow \varepsilon'_i$. We note that due to our training setup (see our Github repository) each client performed local training sequentially, not simultaneously (which would be standard in real world settings), which increased the computation time for full federated training. However, even when dividing the time by the number of clients (5) to get a better estimate, the times for full training on all three datasets are still significantly larger than that of estimating the parameters with FLDP-FL (ACS Income: 284.70s, Glioma: 9.88s, ACS Public Coverage: 143.64s). These results reinforce the discussion in Section 4.4 that using FLDP-FL offers the benefit of being more efficient that full federated re-training. We also point out that the time for full federated re-training will inevitably increase as the federated dataset and model grow larger, or if multiple clients decide to change $\varepsilon_i \rightarrow \varepsilon'_i$, while the time of computing FLDP-FL will remain relatively stable as only the client(s) updating $\varepsilon_i \rightarrow \varepsilon'_i$ must calculate Eq. 14 based on their own local dataset.

Table 3. Time in seconds to perform full federated training with 5 clients on each dataset compared with one client calculating FLDP-FL via Eq. 14.

	Full Fed. Training	FLDP-FL (Eq. 14)
ACS Income	$1423.51_{\pm 12.40}$	$8.24_{\pm 0.38}$
Glioma	$49.41_{\pm 4.13}$	$0.93_{\pm 0.13}$
ACS Public Coverage	$717.32_{\pm 8.08}$	$87.40_{\pm 4.38}$

6 Conclusion

In this work, we proposed Flexible Local Differential Privacy for Federated Learning (FLDP-FL), a technique based on influence functions for local differentially private federated learning which allows clients to change their privacy parameter ε post-training without having to retrain the federated model. Through empirical evaluation on three datasets, we show that FLDP-FL is able to estimate the true parameters that would be obtained if the client's data was re-randomized under their new ε value and federated retraining was repeated. Further, we also show that FLDP-FL is able to support multiple clients changing their ε value after training when the dataset is of sufficient size and can also generate reasonable estimations when the client performing the change owns a large portion of the overall federated dataset. Future work will include utilizing second-order influence functions instead of first-order estimations to see if better loss difference, accuracy difference, and euclidean distance between the estimated and true parameters can be obtained as well as using local differential privacy methods beyond randomized response.

Acknowledgments. This work was supported in part by the NSF under awards 1920920, 1946391, and 2119691, the National Institute of General Medical Sciences of National Institutes of Health under award P20GM139768, and the Arkansas Integrative Metabolic Research Center at the University of Arkansas.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article

References

1. Abadi, M., Chu, A., Goodfellow, I., McMahan, H.B., Mironov, I., Talwar, K., Zhang, L.: Deep learning with differential privacy. In: ACM SIGSAC CCS. pp. 308–318 (2016)
2. Andrew, G., Thakkar, O., McMahan, B., Ramaswamy, S.: Differentially private learning with adaptive clipping. NeurIPS **34**, 17455–17466 (2021)
3. Carey, A.N., Van, M.H., Wu, X.: Evaluating the impact of local differential privacy on utility loss via influence functions. In: IJCNN. pp. 1–10. IEEE (2024)
4. Cook, R.D.: Assessment of local influence. J. R. Stat. Soc., B: Stat. Methodol. **48**(2), 133–155 (1986)
5. Ding, F., Hardt, M., Miller, J., Schmidt, L.: Retiring adult: New datasets for fair machine learning. NeurIPS **34**, 6478–6490 (2021)

6. Dwork, C., Lei, J.: Differential privacy and robust statistics. In: 41st ACM STOC. pp. 371–380 (2009)
7. Fu, J., Hong, Y., Ling, X., Wang, L., Ran, X., Sun, Z., Wang, W.H., Chen, Z., Cao, Y.: Differentially private federated learning: A systematic review. arXiv:2405.08299 (2024)
8. Hu, R., Guo, Y., Li, H., Pei, Q., Gong, Y.: Personalized federated learning with differential privacy. IEEE IOTJ **7**(10), 9530–9539 (2020)
9. Huang, J., Hong, C., Liu, Y., Chen, L.Y., Roos, S.: Maverick matters: Client contribution and selection in federated learning. In: PAKDD. pp. 269–282. Springer (2023)
10. Kairouz, P., McMahan, H.B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A.N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al.: Advances and open problems in federated learning. Foundations and trends® in machine learning **14**(1–2), 1–210 (2021)
11. Kang, Y., Liu, Y., Ding, L., Liu, X., Tong, X., Wang, W.: Differentially private erm based on data perturbation. arXiv:2002.08578 (2020)
12. Kasiviswanathan, S.P., Lee, H.K., Nissim, K., Raskhodnikova, S., Smith, A.: What can we learn privately? SICOMP **40**(3), 793–826 (2011)
13. Koh, P.W., Liang, P.: Understanding black-box predictions via influence functions. In: ICML. pp. 1885–1894. PMLR (2017)
14. Koh, P.W.W., Ang, K.S., Teo, H., Liang, P.S.: On the accuracy of influence functions for measuring group effects. NeurIPS **32** (2019)
15. Li, A., Zhang, L., Wang, J., Han, F., Li, X.Y.: Privacy-preserving efficient federated-learning model debugging. IEEE TPDS **33**(10), 2291–2303 (2021)
16. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: AISTATS. pp. 1273–1282. PMLR (2017)
17. McMahan, H.B., Ramage, D., Talwar, K., Zhang, L.: Learning differentially private recurrent language models. arXiv:1710.06963 (2017)
18. Rokvic, L., Danassis, P., Faltings, B.: Privacy-preserving data filtering in federated learning using influence approximation. In: Workshop on Federated Learning: Recent Advances and New Challenges (in Conjunction with NeurIPS 2022) (2022)
19. Tan, Y., Long, G., Jiang, J., Zhang, C.: Influence-oriented personalized federated learning. arXiv:2410.03315 (2024)
20. Tasci, E., Camphausen, K., Krauze, A.V., Zhuge, Y.: Glioma Grading Clinical and Mutation Features. UCI Machine Learning Repository (2022), DOI: https://doi.org/10.24432/C5R62J
21. Truex, S., Liu, L., Chow, K.H., Gursoy, M.E., Wei, W.: Ldp-fed: Federated learning with local differential privacy. In: EdgeSys. pp. 61–66 (2020)
22. Wang, W., Zhang, C., Tian, Z., Yu, S.: Fedu: Federated unlearning via user-side influence approximation forgetting. IEEE TDSC (2024)
23. Wang, Y., Wu, X., Hu, D.: Using randomized response for differential privacy preserving data collection. In: EDBT/ICDT. vol. 1558, pp. 0090–6778 (2016)
24. Wei, K., Li, J., Ding, M., Ma, C., Yang, H.H., Farokhi, F., Jin, S., Quek, T.Q., Poor, H.V.: Federated learning with differential privacy: Algorithms and performance analysis. IEEE TIFS **15**, 3454–3469 (2020)
25. Wu, C., Zhu, S., Mitra, P.: Federated unlearning with knowledge distillation. arXiv:2201.09441 (2022)
26. Zhang, X., Chen, X., Hong, M., Wu, Z.S., Yi, J.: Understanding clipping for federated learning: Convergence and client-level differential privacy. In: ICML (2022)