Noise-Aware Algorithm for Heterogeneous Differentially Private Federated Learning

Saber Malekmohammadi [1 2] **Yaoliang Yu** [1 2] **Yang Cao** [3]

Abstract

High utility and rigorous data privacy are of the main goals of a federated learning (FL) system, which learns a model from the data distributed among some clients. The latter has been tried to achieve by using differential privacy in FL (DPFL). There is often heterogeneity in clients' privacy requirements, and existing DPFL works either assume uniform privacy requirements for clients or are not applicable when server is not fully trusted (our setting). Furthermore, there is often heterogeneity in batch and/or dataset size of clients, which as shown, results in extra variation in the DP noise level across clients' model updates. With these sources of heterogeneity, straightforward aggregation strategies, e.g., assigning clients' aggregation weights proportional to their privacy parameters (ϵ) will lead to lower utility. We propose Robust-HDP, which efficiently estimates the true noise level in clients' model updates and reduces the noise-level in the aggregated model updates considerably. Robust-HDP improves utility and convergence speed, while being safe to the clients that may maliciously send falsified privacy parameter ϵ to server. Extensive experimental results on multiple datasets and our theoretical analysis confirm the effectiveness of Robust-HDP. Our code can be found here.

1. Introduction

In the presence of sensitive information in the train data, FL algorithms must be able to provide rigorous data privacy guarantees against a potentially curious server or any third party (Hitaj et al., 2017; Rigaki & García, 2020; Wang et al., 2019b; Zhu et al., 2019; Geiping et al., 2020). Differential Privacy (Dwork et al., 2006b;a; Dwork, 2011; Dwork & Roth, 2014) has been used in DPFL systems to achieve such formal privacy guarantees. When there is a trusted server in the system, FL with central differential privacy (CDP), which is operated by the server by adding controlled noise to the aggregation of clients' updates, is a solution (McMahan et al., 2018; Geyer et al., 2017). When there is no trusted server, which is more common, FL with local differential privacy (LDP), where each client randomizes its updates locally is also a solution (Zhao et al., 2021). However, LDP is limited in the sense that achieving privacy while preserving model utility is challenging, due to clients' independent noise additions. Some solutions have been proposed for improving utility in LDP, e.g., using a trusted shuffler system (Liu et al., 2021b; Girgis et al., 2021), which may be difficult to establish if the server itself is not trusted.

Clients often have heterogeneous privacy preferences coming from their varying privacy policies. Furthermore, dataset size usually varies a lot across clients. Additionally, depending on their computational budgets, some clients may use relatively smaller batch sizes locally for running DPSGD algorithm (Abadi et al., 2016). As we will show, a small privacy parameter (ϵ) and/or a small batch size lead to a fast increment of the noise level in a client's model update. Existing heterogeneous DPFL works mostly either depend on a trusted server, i.e., CDP (Chathoth et al., 2022; Zhou et al., 2022), or suffer from suboptimal and vulnerable aggregation strategies on an untrusted server (i.e., LDP) based on clients' privacy parameters (Liu et al., 2021a). We consider local heterogeneous DPFL systems with an *untrusted* server and propose an efficient algorithm, which is aware of the noise level in each client's model update. We propose to employ Robust PCA (RPCA) algorithm (Candes et al., 2009) by the untrusted server to estimate the amount of noise in clients' model updates, which we show depends strongly on multiple factors (e.g., their privacy parameter and their batch size ratio), and assign their aggregation weights accordingly. The use of this efficient strategy on the server, which is independent of clients sending any privacy parameters to the server or not, improves model utility and convergence speed while being robust to potential falsifying clients. The

[1] School of Computer Science, University of Waterloo, Waterloo, Canada [2] Vector Institute, Toronto, Canada [3] Department of Computer Science, Tokyo Institute of Technology, Tokyo, Japan. Correspondence to: Saber Malekmohammadi <saber.malekmohammadi@uwaterloo.ca>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

highlights of our contributions are the followings:

- We show the effect of privacy parameter and batch/dataset size on the noise level in clients' updates.
- We propose "Robust-HDP", a noise-aware robust algorithm for local heterogeneous DPFL (untrusted server)
- As the first work assuming heterogeneous dataset sizes, heterogeneous batch sizes, non-uniform and varying aggregation weights and partial participation of clients simultaneously, we prove convergence of our proposed algorithm under mild assumptions on loss functions.
- In various heterogeneity scenarios across clients, we show that Robust-HDP improves utility and convergence speed while respecting clients' privacy.

2. Related work

Differential privacy. In this work, we use the following definition of differential privacy:

Definition 2.1 ((ϵ, δ)-DP (Dwork et al., 2006a)). A randomized mechanism $\mathcal{M} : \mathcal{D} \rightarrow \mathcal{R}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ satisfies (ϵ, δ)-DP if for any two adjacent inputs $d, d' \in \mathcal{D}$, which differ only by a single record, and for any measurable subset of outputs $\mathcal{S} \subseteq \mathcal{R}$ it holds that

$$\mathtt{Pr}[\mathcal{M}(d) \in \mathcal{S}] \leq e^{\epsilon} \mathtt{Pr}[\mathcal{M}(d') \in \mathcal{S}] + \delta.$$

Gaussian mechanism, which randomizes the output of a non-private computation f on a dataset d as $\mathbf{G}_{\sigma} f(d) \triangleq f(d) + \mathcal{N}(\mathbf{0}, \sigma^2)$, provides (ϵ, δ)-DP. The variance of the noise, σ^2, is calibrated to the sensitivity of f, i.e., the maximum amount of change in its output (measured in ℓ_2 norm) on two neighboring datasets d and d'. Gaussian mechanism has been used in DPSGD algorithm (Abadi et al., 2016) for private ML to randomize intermediate data-dependent computations, e.g., gradients. Some prior works (Gur-Ari et al., 2018) found that stochastic gradients stay in a low-dimensional space during training with Stochastic Gradient Descent (SGD). Inspired by this, Zhou et al. (2021) proposed projection-based variant of the DPSGD (Abadi et al., 2016) algorithm (projected DPSGD), which improves utility by removing the unnecessary noise from noisy batch gradients by projecting them on a linear subspace obtained from a public dataset. Personalized DP (PDP), which specifies a separate privacy parameter ϵ for each data sample in a dataset, was used for centralized settings in (Alaggan et al., 2017; Jorgensen et al., 2015; Huang et al., 2020; Kotsogiannis et al., 2020; Yu et al., 2023), followed by some recent works in (Boenisch et al., 2023; Heo et al., 2023). Another similar work in (Niu et al., 2020) proposed "Utility Aware Exponential Mechanism" (UPEM) to pursue higher utility while achieving PDP. In the same direction of improving utility, Shi et al. (2021) proposed "Selective DP" for improving utility by leveraging the fact that private information in natural language is sparse.

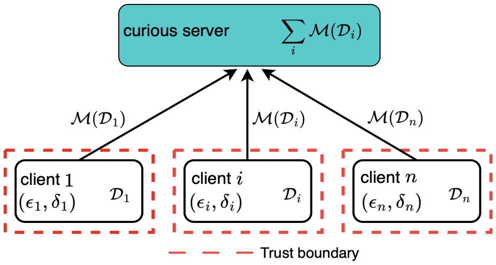

Figure 1. Security model in local heterogeneous DPFL, where client i has local train data $\mathcal{D}_i$ and privacy parameters (ϵ_i, δ_i), and does not trust any external parties.

Heterogeneous DPFL. Assuming the existence of a *trusted* server, Chathoth et al. (2022) proposed cohort-level privacy with privacy and data heterogeneity across cohorts using ϵ-DP definition (Definition 2.1 with $\delta = 0$). Also, the work in (Zhou et al., 2022), adapted the non-uniform sampling idea of (Jorgensen et al., 2015) to the FL settings with a *trusted* server to get client-level DPFL (i.e., d and d' differ by one client's whole data) against membership inference attacks (Rigaki & García, 2020; Wang et al., 2019b). In contrast, we consider *untrusted* servers.

The output of an algorithm $\mathcal{M}$, in the sense of Definition 2.1, is all the information that the untrusted server, which we want to protect against, observes. We consider heterogeneous local model of DPFL (Figure 1), where each client i has its own desired privacy parameters (ϵ_i, δ_i), and sends data-dependent computation results $\mathcal{M}(\mathcal{D}_i)$ (i.e., model updates) to the server. Also, in the context of Definition 2.1, the notion of neighboring datasets that we consider in this work, refers to pair of federated datasets $d = \{\mathcal{D}_1, \cdots, \mathcal{D}_n\}$ and $d' = \{\mathcal{D}_1, \cdots, \mathcal{D}_n\}$, differing by one data point of one client (i.e., *record-level* DPFL). Liu et al. (2021a) adapted a projection-based approach, similar to that of projected DPSGD (Zhou et al., 2021), to the heterogeneous DPFL setting to propose PFA and improve utility. Although assuming an untrusted server, their proposed algorithm relies on the assumption that the server knows the clients' *"true"* privacy parameters $\{(\epsilon_i, \delta_i)\}$ and uses them to cluster clients to "public" (those with larger privacy parameters) and "private". As such, as we show, PFA is extremely vulnerable to when clients share a falsified value of their privacy parameters (often larger than their true values) with server. Also, they used aggregation strategy $w_i \propto \epsilon_i$ on server for PFA and another algorithm called WeiAvg (see Table 1). As we will show, *even if the server knows clients' true privacy parameters, this information is not a perfect indication of the "true" noise level in their model updates, especially with heterogeneous privacy parameters and batch/dataset sizes.*

Table 1. Features of different heterogeneous `DPFL` algorithms. ×: needed at server, ✓: not needed.

algorithm	**aggregation strategy**	$\{\epsilon_i\}_{i=1}^n$	**clients clustering**	**PCA on clients updates**
WeiAvg (**Liu et al., 2021a, Alg. 2**)	$w_i \propto \epsilon_i$	×	×	✓
PFA (**Liu et al., 2021a**)	$w_i \propto \epsilon_i$	×	×	×
DPFedAvg (**Noble et al., 2021**)	$w_i \propto N_i$	✓	✓	✓
minimum ϵ (**Liu et al., 2021a**)	$w_i \propto N_i$	×	✓	✓
Robust-HDP (**Alg. 1**)	$w_i \propto \frac{1}{\sigma_i^2}$	✓	✓	✓

The current state of the art in local heterogeneous `DPFL` calls for a robust algorithm that takes all the mentioned potential sources of heterogeneity across clients into account and achieves high utility and data privacy simultaneously.

3. The Robust-HDP algorithm for heterogeneous DPFL

In this section, we will devise a new heterogeneous `DPFL` algorithm, and we first explain the intuitions behind it. Used notations are explained in Table 2 and Appendix A in details. At the t-th gradient update step on a current model $\boldsymbol{\theta}$, client i computes the following noisy batch gradient:

$$\tilde{g}_i(\boldsymbol{\theta}) = \frac{1}{b_i}\left[\Big(\sum_{j\in\mathcal{B}_i^t}\bar{g}_{ij}(\boldsymbol{\theta})\Big) + \mathcal{N}(\mathbf{0}, \sigma^2_{i,\mathrm{DP}}\mathbb{I}_p)\right], \quad (1)$$

where $\bar{g}_{ij}(\boldsymbol{\theta}) = \mathtt{clip}(\nabla\ell(h(x_{ij},\boldsymbol{\theta}), y_{ij}), c)$, and c is a clipping threshold. For a given vector $\mathbf{v}$, $\mathtt{clip}(\mathbf{v}, c) = \min\{\|\mathbf{v}\|, c\} \cdot \frac{\mathbf{v}}{\|\mathbf{v}\|}$. Also, $\sigma_{i,\mathrm{DP}} = c \cdot z(\epsilon_i, \delta_i, q_i, K_i, E)$: knowing E (global communication rounds), client i can compute $z(\epsilon_i, \delta_i, q_i, K_i, E)$ locally, which is the noise scale that it should use locally for DPSGD in order to achieve $(\epsilon_i, \delta_i)-\mathtt{DP}$ with respect to $\mathcal{D}_i$ at the end of E global rounds. This can be done by client i using a privacy accountant, e.g., the moments accountant (Abadi et al., 2016). Therefore, *depending on its privacy preference* (ϵ_i, δ_i), each client i computes its required noise scale z, runs DPSGD locally and sends its noisy model updates to the server at the end of each round. Now, an important question is that what is an efficient aggregation strategy for the server to aggregate the clients' noisy model updates? Intuitively, the server has to pay more attention to the less noisy updates. The challenge is that the server knows neither the noise added by each client i nor its amount. To answer the above question, we first analyze the behavior of the noise level in clients' batch gradients in Section 3.1, which is used in Section 3.2, for a similar analysis of clients' uploaded model updates. The result of this analysis is an idea we propose for the server to estimate the noise amount in each model update, which leads to an efficient aggregation strategy in Section 3.4.

Table 2. Used notations (also see Appendix A)

n	number of clients, which are indexed by i
x_{ij}, y_{ij}	j-th data point of client i and its label
$\mathcal{D}_i, N_i$	local train set of client i and its size
$\mathcal{B}_i^t$	the train data batch used by client i at the t-th gradient update
b_i	batch size of client i: $\lvert\mathcal{B}_i^t\rvert = b_i$
q_i	batch size ratio of client i: $q_i = \frac{b_i}{N_i}$
ϵ_i, δ_i	client i's desired `DP` privacy parameters
E	total number of global communication rounds in the `DPFL` system, indexed by e
$\mathcal{S}^e$	set of participating clients in round e
$\boldsymbol{\theta}^e$	global model parameter, which has size p, at the beginning of global round e
K_i	number of local train epochs performed by client i during each global round e
E_i	number of batch gradient updates of client i during each global round e: $E_i = K_i \cdot \lceil\frac{N_i}{b_i}\rceil$
h	predictor function, e.g., CNN model, with parameter $\boldsymbol{\theta}$
ℓ	cross entropy loss
$\sigma^2_{i,\tilde{g}}$	variance of the noisy stochastic batch gradient $\tilde{g}_i(\boldsymbol{\theta})$ of client i
σ_i^2	conditional variance of the noisy model update $\Delta\tilde{\boldsymbol{\theta}}_i^e$ of client i: $\mathtt{Var}(\Delta\tilde{\boldsymbol{\theta}}_i^e \vert \boldsymbol{\theta}^e)$

3.1. Noise level in clients' DP batch gradients

We consider two cases, which are easier to analyze. Our analysis gives us an understanding of the parameters affecting the noise level in clients' batch gradients. Depending on the value of the used clipping threshold c at the t-th gradient update step, we consider two general indicative cases:

1. Effective clipping threshold for all samples: in this case, from Equation (1), we have:

$$\mathbb{E}[\tilde{g}_i(\boldsymbol{\theta})] = \frac{1}{b_i}\sum_{j\in\mathcal{B}_i^t}\mathbb{E}[\bar{g}_{ij}(\boldsymbol{\theta})] = \frac{1}{b_i}\sum_{j\in\mathcal{B}_i^t}G_i(\boldsymbol{\theta}) = G_i(\boldsymbol{\theta}), \quad (2)$$

where the expectation is with respect to the stochasticity of gradients and we have assumed that $E[\bar{g}_{ij}(\boldsymbol{\theta})]$ is the same

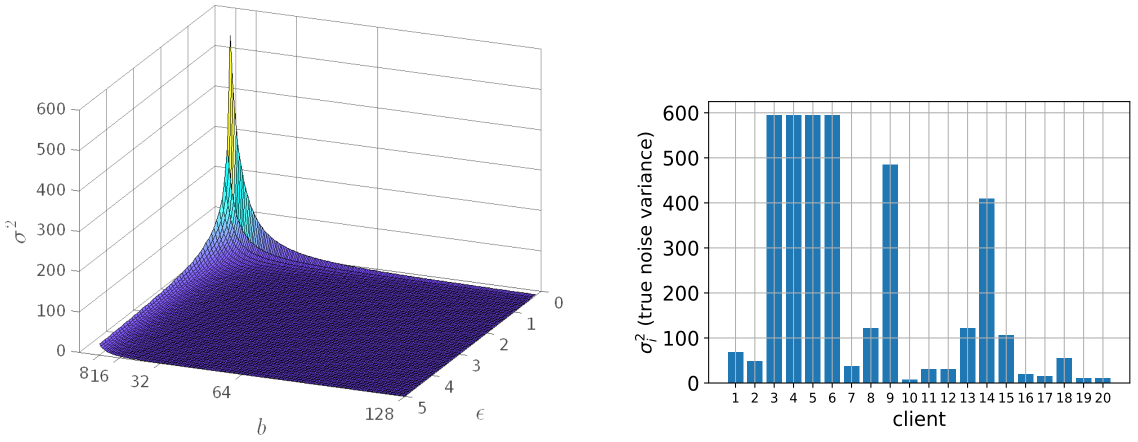

Figure 2. Left: 3D plot of noise variance σ_i^2 of a client i (Equation (7) with $K_i = 1, N_i = 2400, \eta_l = 0.01, c = 3, p = 28939$) based on b_i and the privacy budget ϵ_i. Right: the noise variances $\{\sigma_i^2\}_{i=1}^n$ in a $\mathtt{DPFL}$ system with $n = 20$ clients, where $\{(\epsilon_i, b_i)\}_{i=1}^n$ are randomly selected for each client. It clearly shows an approximately *sparse pattern* (14 of the clients have much smaller noise variance than the other 6). Each bar plot in the right figure corresponds to a point in the left figure.

for all j and is denoted by $G_i(\boldsymbol{\theta})$. Now, for an arbitrary random variable $\mathbf{v} = (v_1, \dots, v_p)^\top \in \mathbb{R}^{p\times 1}$, we define $\mathtt{Var}(\mathbf{v}) := \sum_{j=1}^{p} \mathbb{E}[(v_j - \mathbb{E}[v_j])^2]$, i.e., variance of $\mathbf{v}$ is the sum of the variances of its elements. Then, the variance of the noisy stochastic gradient in Equation (1), which is also random, can be computed as (see Appendix C):

$$\begin{aligned}\sigma_{i,\tilde{g}}^2 &:= \mathtt{Var}(\tilde{g}_i(\boldsymbol{\theta})) \\ &= \frac{c^2 - \left\|G_i(\boldsymbol{\theta})\right\|^2}{b_i} + \frac{pc^2 z^2(\epsilon_i, \delta_i, q_i, K_i, E)}{b_i^2} \\ &\approx \frac{pc^2 z^2(\epsilon_i, \delta_i, q_i, K_i, E)}{b_i^2}, \end{aligned} \tag{3}$$

where, the estimation is valid because $p \gg 1$. For instance, $p \approx 2 \times 10^7$ for ResNet-34 for CIFAR100, and $c = 3$.

2. Ineffective clipping threshold for all samples: in this case, we have a noisy version of the batch gradient $g_i(\boldsymbol{\theta}) = \frac{1}{b_i}\sum_{j\in\mathcal{B}_i^t} g_{ij}(\boldsymbol{\theta})$, which is unbiased with variance bounded by $\sigma_{i,g}^2$ (see Assumption D.1). Hence:

$$\mathbb{E}[\tilde{g}_i(\boldsymbol{\theta})] = \mathbb{E}[g_i(\boldsymbol{\theta})] = \nabla f_i(\boldsymbol{\theta}), \tag{4}$$

$$\begin{aligned}\sigma_{i,\tilde{g}}^2 &= \mathtt{Var}(\tilde{g}_i(\boldsymbol{\theta})) = \mathtt{Var}(g_i(\boldsymbol{\theta})) + \frac{p\sigma_{i,\mathtt{DP}}^2}{b_i^2} \\ &\leq \sigma_{i,g}^2 + \frac{pc^2 z^2(\epsilon_i, \delta_i, q_i, K_i, E)}{b_i^2}. \end{aligned} \tag{5}$$

z is a sub-linearly increasing function of q_i (and equivalently b_i: see Theorem 3.1 and Figure 8 in the appendix). It is also clear that z is a decreasing function of ϵ_i and δ_i. Hence, *$\sigma_{i,\tilde{g}}^2$ is a decreasing function of b_i (batch size), N_i (dataset size) and ϵ_i, and also an increasing function of q_i (batch size ratio).*

3.2. Noise level in clients' $\mathtt{DP}$ model updates

Having found the parameters affecting $\sigma_{i,\tilde{g}}^2$, we now investigate the parameters affecting the noise level in clients' model updates. During each global communication round e, a participating client i performs $E_i = K_i \cdot \lceil \frac{N_i}{b_i} \rceil = K_i \cdot \lceil \frac{1}{q_i} \rceil$ batch gradient updates locally with step size η_l:

$$\begin{aligned}\Delta\tilde{\boldsymbol{\theta}}_i^e &= \boldsymbol{\theta}_{i,E_i}^e - \boldsymbol{\theta}_{i,0}^e, \\ \boldsymbol{\theta}_{i,k}^e &= \boldsymbol{\theta}_{i,k-1}^e - \eta_l \tilde{g}_i(\boldsymbol{\theta}_{i,k-1}^e),\ k = 1, \dots, E_i, \end{aligned} \tag{6}$$

where $\boldsymbol{\theta}_{i,0}^e = \boldsymbol{\theta}^e$. In each update, it adds a Gaussian noise from $\mathcal{N}(\mathbf{0}, \frac{c^2 z^2(\epsilon_i, \delta_i, q_i, K_i, E)}{b_i^2}\mathbb{I}_p)$ to its batch gradients independently (see Equation (1)). Hence:

$$\sigma_i^2 := \mathtt{Var}(\Delta\tilde{\boldsymbol{\theta}}_i^e | \boldsymbol{\theta}^e) = K_i \cdot \lceil \frac{1}{q_i} \rceil \cdot \eta_l^2 \cdot \sigma_{i,\tilde{g}}^2, \tag{7}$$

where $\sigma_{i,\tilde{g}}^2$ was computed in Equation (3) and Equation (5) for two general indicative cases. This means that σ_i^2 heavily depends on b_i (e.g., when clipping is effective, b_i appears with power 3 in denominator. Recall $\frac{1}{q_i} = \frac{N_i}{b_i}$). Hence, σ_i^2 decreases quickly when b_i increases. Similarly, σ_i^2 is a non-linearly decreasing function of ϵ_i (see Figure 2, left). However, note that N_i and q_i appear twice in Equation (7) with opposing effects. This makes the variation of σ_i^2 with N_i and q_i small (explained in details in Appendix G.3). An important message of these important understandings is that *ϵ_i is not the only parameter of client i that determines σ_i^2.*

3.3. Optimum aggregation strategy

Assuming the set of participating clients $\mathcal{S}^e$ in round e, we have to solve the following problem to minimize the total noise after the aggregation at the end of this round:

$$\begin{aligned}\min_{w_i \geq 0} \quad & \texttt{Var}\Big(\sum_{i\in\mathcal{S}^e} w_i \Delta\tilde{\boldsymbol{\theta}}_i^e \big| \boldsymbol{\theta}^e\Big) = \sum_{i\in\mathcal{S}^e} {w_i}^2 \sigma_i^2, \\ & \texttt{s.t.} \sum_{i\in\mathcal{S}^e} w_i = 1, \end{aligned} \tag{8}$$

which has a unique solution $w_i^* \propto \frac{1}{\sigma_i^2}$. Hence, the optimum aggregation strategy weights clients directly based on $\{\sigma_i^2\}_{i=1}^n$, which as shown, *not only depends on* $\{\epsilon_i\}_{i=1}^n$ *non-linearly, but it also depends on* $\{b_i\}_{i=1}^n$ *and* $\{N_i\}_{i=1}^n$. This point makes the aggregation strategy $w_i \propto \epsilon_i$ of PFA and WeiAvg algorithms (Liu et al., 2021a) suboptimal, let alone its vulnerability to a client i sharing a falsified $\epsilon_i' > \epsilon_i$ with the server to either attack the system, to get a larger aggregation weight, or to get a larger payment from a server which incentivizes participation by payment to clients (Donahue & Kleinberg, 2021; Karimireddy et al., 2022; Fallah et al., 2023; Kang et al., 2023) (as a larger ϵ_i means a more exploitable data from client i). The same vulnerability discussion applies to the clustering of clients based on their shared privacy parameter ϵ (used in PFA). Having these shortcomings of the existing algorithms as a motivation, how can we implement the optimum aggregation strategy when the untrusted server does not have any idea of the clients noise addition mechanisms and $\{\sigma_i^2\}_{i=1}^n$? We next propose our idea for estimating $\{\sigma_i^2\}_{i=1}^n$ and $\{w_i^*\}_{i=1}^n$.

3.4. Description of Robust-HDP algorithm

Assuming a DPFL system with n clients and full participation of clients for simplicity, at the end of each global round e, the server gets the matrix $\mathbf{M} := [\Delta\tilde{\boldsymbol{\theta}}_1^e|\dots|\Delta\tilde{\boldsymbol{\theta}}_n^e]$. Assuming an *i.i.d* or moderately heterogeneous data split, and based on the findings in (Gur-Ari et al., 2018; Zhou et al., 2021), we would expect $\mathbf{M}$ to have a low rank if there was no DP/stochastic noise in $\{\Delta\tilde{\boldsymbol{\theta}}_i^e\}_{i=1}^n$. So we can think about writing $\mathbf{M}$ as the summation of an underlying low-rank matrix $\mathbf{L}$ and a noise matrix $\mathbf{S}$:

$$\mathbf{M} = [\Delta\tilde{\boldsymbol{\theta}}_1^e|\dots|\Delta\tilde{\boldsymbol{\theta}}_n^e] = \mathbf{L} + \mathbf{S}.$$

If the matrix $\mathbf{S}$ is sparse, not only can such a decomposition problem be solved using RPCA, it can be solved by a very convenient convex optimization program called *Principal Component Pursuit* (Algorithm 3 in the appendix) without imposing much computational overhead to the server (Candes et al., 2009). Surprisingly, the entries in $\mathbf{S}$ can have arbitrarily large magnitudes. Theoretically, this is guaranteed to work even if *rank*$(\mathbf{L}) \in \mathcal{O}(n/(\log p)^2)$, i.e., the rank of $\mathbf{L}$ grows almost linearly in n (see Theorem 1.1 in (Candes et al., 2009)). Hence, we expect to be able to do such a decomposition as long as we have a moderately heterogeneous data distribution across a large enough number of clients (also, see Appendix H for detailed discussion on data heterogeneity, further experiments, and future directions). Hence, $\mathbf{L}$ will be a low rank matrix, estimating the "true" values of clients' updates and $\mathbf{S}$ will capture the noises in clients model updates $\{\Delta\tilde{\boldsymbol{\theta}}_i^e\}_{i=1}^n$ induced by two sources: DP additive Gaussian noise and batch gradients stochastic noise. Therefore, we can use $\hat{\sigma}_i^2 := \|\mathbf{S}_{:,i}\|_2^2$ ($\mathbf{S}_{:,i}$ is the i-th column of $\mathbf{S}$, corresponding to client i) as an estimate of σ_i^2 (Equation (7)). Indeed, we observed such approximately sparse pattern for $\mathbf{S}$ in Figure 2 (right), where each barplot corresponds to the ℓ_2 norm of one column of $\mathbf{S}$. Thus, according to Equation (8), we assign the aggregation weights as $w_i^e = \frac{1/\hat{\sigma}_i^2}{\sum_{j\in\mathcal{S}^e} 1/\hat{\sigma}_j^2}$, where $\hat{\sigma}_i^2 = \|\mathbf{S}_{:,i}\|^2$ (see Algorithm 1). *Interestingly, this estimation is independent of clients' shared* ϵ *parameter values, which makes our* Robust-HDP *optimal, robust and vastly applicable.*

Algorithm 1 Robust-HDP

```
Input: Initial parameter $\boldsymbol{\theta}^0$, batch sizes $\{b_1, \dots, b_n\}$,
       dataset sizes $\{N_1, \dots, N_n\}$, noise scales
       $\{z_1, \dots, z_n\}$, gradient norm bound $c$, local
       epochs $\{K_1, \dots, K_n\}$, global round $E$, number of
       model parameters $p$, privacy accountant PA.
Output: $\boldsymbol{\theta}^E, \{\epsilon_1^E, \dots, \epsilon_n^E\}$
Initialize $\boldsymbol{\theta}_0$ randomly
for $e \in [E]$ do
    sample a set of clients $\mathcal{S}^e \subseteq \{1, \dots, n\}$
    for each client $i \in \mathcal{S}^e$ in parallel do
        $\Delta\tilde{\boldsymbol{\theta}}_i^e \leftarrow$DPSGD ($\boldsymbol{\theta}^e, b_i, N_i, K_i, z_i, c$)
        $\epsilon_i^e \leftarrow$ PA($\frac{b_i}{N_i}, z_i, K_i, e$)
    $\mathbf{M} = [\Delta\tilde{\boldsymbol{\theta}}_1^e|\dots|\Delta\tilde{\boldsymbol{\theta}}_{|\mathcal{S}^e|}^e] \in \mathbb{R}^{p\times|\mathcal{S}^e|}$
    $\mathbf{L}, \mathbf{S}$ = RPCA($\mathbf{M}$)
    for $i \in \mathcal{S}^e$ do
        $w_i^e \leftarrow \frac{1/\|\mathbf{S}_{:,i}\|_2^2}{\sum_{j\in\mathcal{S}_e} 1/\|\mathbf{S}_{:,j}\|_2^2}$
    $\boldsymbol{\theta}^{e+1} \leftarrow \boldsymbol{\theta}^e + \sum_{i\in\mathcal{S}_e} w_i^e \Delta\tilde{\boldsymbol{\theta}}_i^e$
```

3.5. Reliability of Robust-HDP

In order for Robust-HDP to assign the optimum aggregation weights $\{w_i^*\}$, *it suffices to estimate the set* $\{\sigma_i^2\}$ *up to a multiplicative factor*. Assuming participants $\mathcal{S}^e$ in round e, let $s_{i,j}$ in matrix $\mathbf{S}$ represent the true value of noise in the i-th element of $\Delta\tilde{\boldsymbol{\theta}}_j^e$ ($j \in \mathcal{S}^e$). Then, assume that $\mathbf{S}'$ is the matrix computed by Robust-HDP at the server with bounded elements $s'^2_{i,j} \leq U$, where $\mathbb{E}[s'_{i,j}] = rs_{i,j}$, for some constant $r > 0$, and $\mathbb{E}[|s'_{i,j} - rs_{i,j}|^2] \leq \alpha_j^2$ (i.e., on average, Robust-HDP is able to estimate the true noise values $s_{i,j}$ up to a multiplicative factor r by using RPCA). Then, from Hoeffding's inequality, we have:

$$\texttt{Pr}(|\hat{\sigma}_j^2 - (r^2\sigma_j^2 + \alpha_j^2)| > \epsilon) \leq 2e^{\frac{-2p\epsilon^2}{U^2}}, \tag{9}$$

meaning that estimating the entries of $\mathbf{S}$ up to a multiplicative factor r with a small variance is enough for Robust-HDP to estimate $\{\sigma_i^2\}$ up to a multiplicative factor r^2 with high probability. This probability increases with the number of model parameters p exponentially: the p noise elements of $\mathbf{S}_{:,i}$ are *i.i.d*, and larger p means having more samples from the same distribution to estimate its variance (see also Theorem 1.1 in (Candes et al., 2009)). Also, $w_j \propto \frac{1}{\hat{\sigma}_j^2} \approx \frac{1}{r^2\sigma_j^2+\alpha_j^2}$. Hence, as $\sigma_j^2 \gg 1$ (it is the noise variance in the whole model update $\Delta\tilde{\boldsymbol{\theta}}_j^e$. See the values in Figure 2, right), a small deviation α_j^2 from $r^2\sigma_j^2$ still results in aggregation weights close to the optimum weights $\{w_i^*\}$.

3.6. Scalability of Robust-HDP with the number of model parameters p

The computation time (precision) of RPCA algorithm increases (decreases) when the number of model parameters p grows. As such, in order to make the Robust-HDP scalable for large models, we perform the noise estimation of Robust-HDP on sub-matrices of $\mathbf{M}$ with smaller rows:

$$\begin{aligned}
\mathbf{M}_1 &= \mathbf{M}[0:p'-1,:] = \mathbf{L}_1 + \mathbf{S}_1\\
\mathbf{M}_2 &= \mathbf{M}[p':2p'-1,:] = \mathbf{L}_2 + \mathbf{S}_2\\
&\dots\\
\mathbf{M}_Q &= \mathbf{M}[p-p':p-1,:] = \mathbf{L}_Q + \mathbf{S}_Q,
\end{aligned}$$

where $Q = \left\lfloor \frac{p}{p'} \right\rfloor$. Then, we get a set of noise variance estimates $\{Q\cdot\hat{\sigma}_i^2\}_{i=1}^n$ from each $\mathbf{S}_j, j\in\{1,\dots,Q\}$. Finally, we use the sets' element-wise average for weight assignment. For instance, for CIFAR10 and CIFAR100, we perform RPCA on sub-matrcies of $\mathbf{M}$ with $p' = 200{,}000$ rows, and average their noise variance estimates. Our experimental results show that this approach, even with $Q = 1$ (i.e., using just $\mathbf{M}_1$), still results in assigning aggregation weights close to the optimum weights $\{w_i^*\}$. This idea makes Robust-HDP scalable to large models with large p.

3.7. Privacy analysis of Robust-HDP

We have the following theorem about DP guarantees of our proposed Robust-HDP algorithm.

Theorem 3.1. *For each client i , there exist constants c_1 and c_2 such that given its number of steps $E\cdot E_i$, for any $\epsilon < c_1 q_i^2 E\cdot E_i$, the output model of* Robust-HDP *satisfies $(\epsilon_i,\delta_i)-\mathtt{DP}$ with respect to $\mathcal{D}_i$ for any $\delta_i > 0$ if $z_i > c_2\frac{q_i\sqrt{E\cdot E_i\cdot\log\frac{1}{\delta_i}}}{\epsilon_i}$, where z_i is the noise scale used by the client i for* DPSGD. *The algorithm also satisfies $(\epsilon_{max},\delta_{max})$-DP, where $(\epsilon_{max},\delta_{max}) = \big(\max(\{\epsilon_i\}_{i=1}^n), \max(\{\delta_i\}_{i=1}^n)\big)$.*

Therefore, the model returned by Robust-HDP is (ϵ_i,δ_i)-DP with respect to $\mathcal{D}_i$, satisfying heterogeneous DPFL.

3.8. The optimization side of Robust-HDP

We assume that $f(\boldsymbol{\theta}) = \sum_{i\in[n]}\lambda_i f_i(\boldsymbol{\theta})$, where $\lambda_i = \frac{N_i}{\sum_i N_i}$, has minimum value f^* and minimizer $\boldsymbol{\theta}^*$. We also make some mild assumptions about the loss functions f_i (see Assumptions D.1 and D.2 in the Appendix). We now analyze the convergence of the Robust-HDP algorithm.

Theorem 3.2 (Robust-HDP)**.** *Assume that Assumptions D.1 and D.2 hold, and for every i, learning rate η_l satisfies: $\eta_l \leq \frac{1}{6\beta E_i}$ and $\eta_l \leq \frac{1}{12\beta\sqrt{(1+\sum_{i=1}^n E_i)\left(\sum_{i=1}^n E_i^4\right)}}$. Then, we have:*

$$\begin{aligned}
&\min_{0\leq e\leq E-1}\mathbb{E}[\|\nabla f(\boldsymbol{\theta}^e)\|^2]\\
&\leq \frac{12}{11E_l^{min}-7}\left(\frac{f(\boldsymbol{\theta}^0)-f^*}{E\eta_l} + \Psi_\sigma + \Psi_p\right), \qquad (10)
\end{aligned}$$

where $E_l^{min} = \min_i E_i$, i.e., the minimum number of local SGD steps across clients. Also, Ψ_p and Ψ_σ are two constants controlling the quality of the final model parameter returned by Robust-HDP*, which are explained in the following.*

Discussion. Our convergence guarantees are quite general: we allow for partial participation, heterogeneous number of local steps $\{E_i\}$, non-uniform batch sizes $\{b_i\}$, varying and nonuniform aggregation weights $\{w_i^e\}$. When $\{f_i\}$ are convex, Robsut-HDP solution converges to a neighborhood of the optimal solution. The term Ψ_σ decreases when data split across clients is more *i.i.d*, and variance of mini-batch gradients $\{\sigma_{i,\tilde{g}}^2\}$ decrease (e.g., when clients are less privacy sensitive). Similarly, Ψ_p decreases when clients participate more often, and the set of local steps $\{E_i\}$ is more uniform (e.g., clients have similar dataset sizes and batch sizes). Also, smaller local steps $\{E_i\}$, which can be achieved by having smaller local epochs $\{K_i\}$ and larger batch sizes $\{b_i\}$, result in reduction of both Ψ_p and Ψ_σ, and higher quality solutions (Malekmohammadi et al., 2023). Compared to the results in previous DPFL works, we have the most general results with more realistic assumptions. For instance, (Liu et al., 2021a) (WeiAvg and PFA) assumes uniform number of local SGD updates for all clients, or (Noble et al., 2021) (DPFedAvg) assumes uniform aggregation weights and uniform number of local updates. These assumptions may not be practical in real systems. In a more general view, when we have no DP guarantees, we recover the results for the simple FedAvg algorithm (Zhang et al., 2023). When we additionally have $\sigma = 0$ (i.e., FedAvg on *i.i.d* data), our results are the same as those of SGD (Ghadimi & Lan, 2013):

$$\min_e \mathbb{E}[\|\nabla f(\boldsymbol{\theta}^e)\|^2] \leq \frac{12}{11E_l^{\min}-7}\frac{f(\boldsymbol{\theta}^0)-f^*}{E\eta_l} + \mathcal{O}(\eta_l),$$

which shows convergence rate $\frac{1}{\sqrt{E}}$ with $\eta_l = \mathcal{O}(\frac{1}{\sqrt{E}})$.

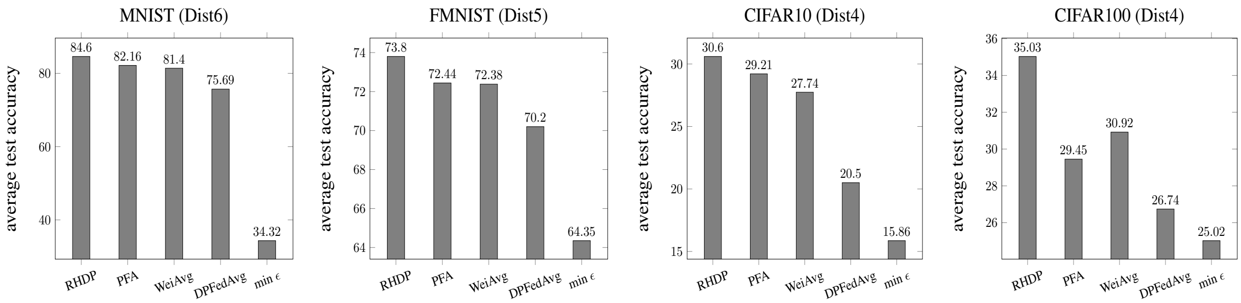

Figure 3. Comparison of average test accuracy between studied algorithms. See Tables 12 to 15 in the appendix for detailed results.

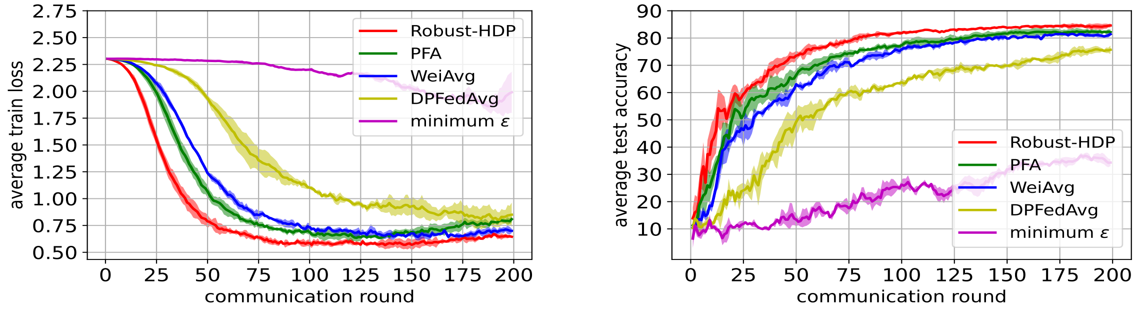

Figure 4. Convergence speed comparison on MNIST and Dist6. Minimum ϵ algorithm diverged in 1 out of 3 trials.

4. Experiments

See Appendix B for details of experimental setup and hyperparameter tuning used for evaluation of algorithms.

4.1. Experimental Setup

Datasets, models and baseline algorithms: We evaluate our proposed method on four benchamrk datasets: MNIST (Deng, 2012), FMNIST (Xiao et al., 2017) and CIFAR10/100 (Krizhevsky, 2009) using CNN-based models. Also, we compare four baseline algorithms: 1. WeiAvg (Liu et al., 2021a) 2. PFA (Liu et al., 2021a) 3. DPFedAvg (Noble et al., 2021) 4. minimum ϵ.

Privacy preference and batch size heterogeneity: We consider an FL setting with 20 clients as explained in Appendix B.1, which results in homogeneous $\{N_i\}_{i=1}^n$. We also assume full participation and one local epoch for each client ($K_i = 1$ for all i). Batch size heterogeneity leads to heterogeneity in the number of local steps $\{E_i\}_{i=1}^n$. We sample $\{\epsilon_i\}_{i=1}^n$ from a set of distributions, as shown in Table 7 in the Appendix. We also sample batch sizes $\{b_i\}_{i=1}^n$ uniformly from $\{16, 32, 64, 128\}$. Therefore, we consider heterogeneous $\{\epsilon_i\}_{i=1}^n$, heterogeneous $\{b_i\}_{i=1}^n$ and uniform $\{N_i\}_{i=1}^n$ in this section. We have also considered various other heterogeneity scenarios for clients and more experimental results are reported in Appendix G and H.

4.2. Experimental Results

In this section, we investigate five main research questions about Robust-HDP, as follows.

RQ1: How do various heterogeneous DPFL algorithms affect the system utility? In Fig. 3, we have done a comparison in terms of the average test accuracy across clients. We observe that Robust-HDP outperforms the baselines (see tables 12 to 15 in the appendix for detailed results). It achieves higher system utility by using an efficient aggregation strategy, where it assigns smaller weights to the model updates that are indeed more noisy and minimizes the noise level in the aggregation of clients' model updates. The aggregation strategy of PFA and WeiAvg is sub-optimal, as it can not take the batch size heterogeneity and privacy parameter heterogeneity into account simultaneously.

RQ2: How does Robust-HDP improve convergence speed during training? We have also compared different algorithms based on their convergence speed in Figure 4. While the baseline algorithms suffer from high levels of noise in the aggregated model update $\sum_{i \in \mathcal{S}^e} w_i^e \Delta\tilde{\boldsymbol{\theta}}_i^e$ (see Table 3), Robust-HDP enjoys its efficient noise minimization, which performs very close to the optimum aggregation strategy, and not only results in faster convergence but also improves utility. In contrast, based on our experiments, the baseline algorithms have to use smaller learning rates to avoid divergence of their training optimization. Note that fast convergence of DPFL algorithms is indeed important, as the privacy budgets of participating clients does not let the server to run the federated training for more rounds.

RQ3: Is Robust-HDP indeed Robust? In Fig. 5, we compare Robust-HDP with others based on clients' desired privacy level and number of clients. As clients become more privacy sensitive, they send more noisy updates to the server,

Table 3. The average *per parameter* noise variance (Equation (7) and Equation (3)) normalized by used learning rate ($\frac{\sum_{i=1}^{n} w_i^{e\,2}\sigma_i^2}{p\eta_l^2}$) in the aggregated model update ($\sum_{i=1}^{n} w_i^e \Delta\tilde{\boldsymbol{\theta}}_i^e$) at the end of first round $(e = 1)$ on FMNIST with $E = 200$. Due to the projection used in PFA, computation of its noise variance was not possible. Results for Robust-HDP are shown with std variation across three experiments.

dist alg	Dist1	Dist2	Dist3	Dist4	Dist5	Dist6	Dist7	Dist8	Dist9
WeiAvg	1.02	1.89	0.92	3.22	4.58	28.29	9.85	48.15	34.91
DPFedAvg	1.27	16.94	16.28	26.87	25.64	70.71	18.50	85.70	43.20
minimum ϵ	4.68	103.91	103.91	127.18	103.91	1868.45	74.41	241.37	87.15
Robust-HDP	**0.27** ±1.9e-5	**0.47** ±5.7e-5	**0.07** ±2.9e-6	**0.64** ±1.0e-4	**0.39** ±9.3e-6	**7.62** ±1.3e-3	**2.25** ±5.5e-5	**13.86** ±9.9e-4	**5.95** ±2.7e-4
Oracle (Eq. 8)	0.27	0.47	0.07	0.64	0.39	7.60	2.25	13.81	5.93

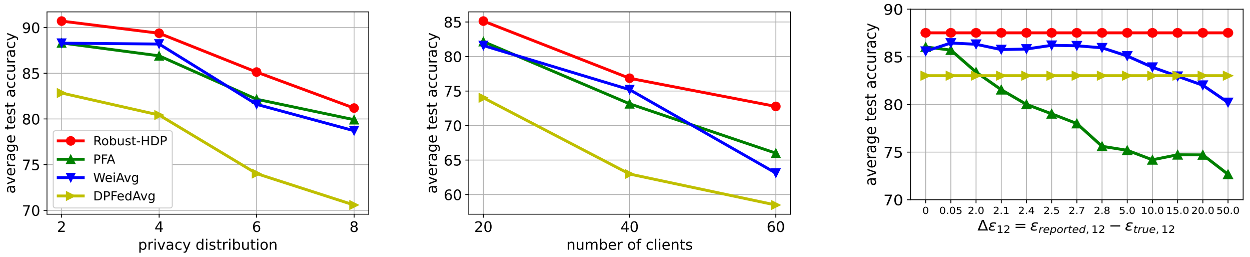

Figure 5. Performance comparison on MNIST. **Left:** effect of clients desired privacy on utility (detailed results in Table 16) **Middle:** effect of number of existing clients (privacy parameters of clients are sampled from Dist6) on utility (detailed results in Table 17) **Right:** Robustness of Robust-HDP when a random client (client 12 with a moderate ϵ value of 0.95) sends falsified version of its ϵ to the server for *aggregation* (privacy parameters of other clients are sampled from Dist5). WeiAvg and PFA are much vulnerable to this falsification.

making convergence to better solutions harder. Robust-HDP shows the highest robustness to the larger noise in clients' updates and achieves the highest utility, especially in more privacy sensitive scenarios, e.g., Dist8. Also, we observe that it achieves the highest system utility when the number of clients in the system increases. Furthermore, it is completely safe in scenarios that some clients report a falsified privacy parameter to the server (Figure 5, right).

RQ4: How accurate Robust-HDP is in estimating $\{w_i^*\}$? Figure 6 compares the weight assignment of Robust-HDP with the optimum assignment (computed from Equations 8) for CIFAR10 dataset and Dist2. As the model used for CIFAR10 is relatively large (with $p \approx 11 \times 10^6$), we have used the approximation method in Section 3.6 (with $Q = 1$ and $p' = 2 \times 10^5$). Figure 6 has sorted clients based on their privacy parameter ϵ in ascending order. WeiAvg and PFA assign smaller weights to more privacy sensitive clients, while Robust-HDP assigns smaller weights to the clients with less noisy model updates.

We have also studied the effect of parameter p', on the precision of the aggregation weights returned by Robust-HDP. In Figure 7 and for CIFAR10, we have shown the increasing precision of the weights returned by Robust-HDP when p' grows. The larger p' gets, the more samples we have

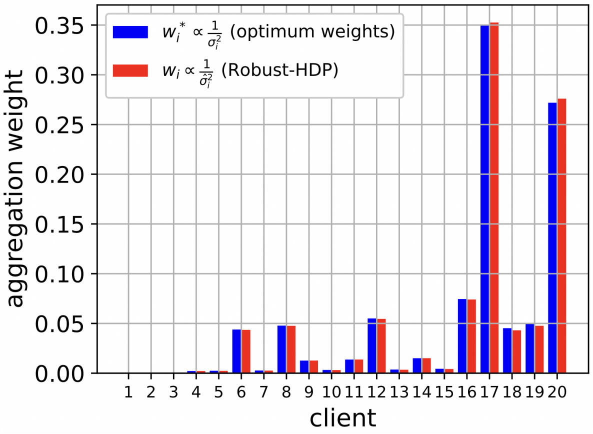

Figure 6. Precision of Robust-HDP (red) compared to oracle optimum strategy (blue) for CIFAR10 and Dist2, when using the approximation method in Section 3.6 with $Q = 1$ and $p' = 2 \times 10^5$.

for estimating the noise variance in clients' model updates, hence more precise weight assignments. As explained in Section 3.6, when p is already large, we also avoid using too large values for p', as the main point of Section 3.6 was to feed a matrix with smaller number of rows to RPCA to avoid its low precision and high computation time when the number of rows (p) in the original input matrix $\mathbf{M}$ is large.

Table 4. Comparison of different algorithms (on MNIST, $E = 200$) with **heterogeneous** data split (maximum 8 labels per client) and **60 clients** in the system all using **uniform batch size 128**.

distr alg	Dist1	Dist2	Dist3	Dist4	Dist5	Dist6	Dist7	Dist8	Dist9
WeiAvg (Liu et al., 2021a)	81.14	81.21	**84.44**	71.64	81.45	72.27	80.55	71.28	72.07
PFA (Liu et al., 2021a)	81.29	77.40	80.65	74.45	81.89	64.48	**81.40**	**73.39**	72.97
DPFedAvg (Noble et al., 2021)	82.61	74.32	83.12	70.8	81.34	66.51	76.10	65.16	73.03
Robust-HDP	**84.84**	**82.78**	80.78	**78.91**	**81.66**	**72.30**	79.32	70.68	**73.82**

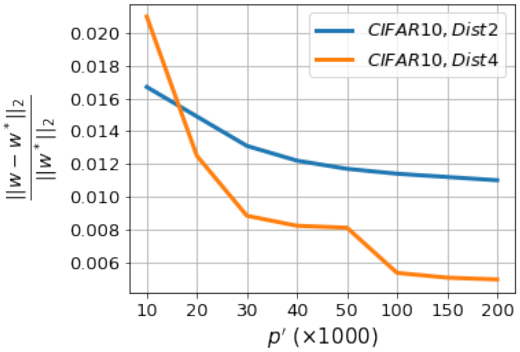

Figure 7. Effect of the parameter p', used in the approximation method explained in Section 3.6, on the precision of the weights returned by Robust-HDP for CIFAR10 (with $p \approx 11 \times 10^6$).

RQ5: What is the effect of data heterogeneity across clients on the performance of Robust-HDP?

So far we assumed an *i.i.d* data distribution across clients. What if the data distribution is moderately/highly heterogeneous? Assuming full participation of clients in round e, in order to have a useful RPCA decomposition $\mathbf{M} = [\Delta\tilde{\boldsymbol{\theta}}_1^e| \dots |\Delta\tilde{\boldsymbol{\theta}}_n^e] = \mathbf{L} + \mathbf{S}$ at the end of the round, two conditions should be met (Candes et al., 2009): 1. There should be an underlying low-rank matrix $\mathbf{L}$ in $\mathbf{M}$ 2. The difference between the matrix $\mathbf{L}$ and $\mathbf{M}$, i.e., the noise matrix $\mathbf{S}$, should be (approximately) sparse.

Whether the first condition is met or not mainly depends on how much heterogeneous the data split across clients is. Note that *rank*$(\mathbf{L})$ should be low, and not necessarily close to 1. If we assume that the second condition is met, it was shown in Theorem 1.1 in (Candes et al., 2009) that the decomposition is guaranteed to work even if *rank*$(\mathbf{L}) \in \mathcal{O}(n/(\log p)^2)$, i.e., the rank of $\mathbf{L}$ grows almost linearly in n. *Therefore, even if the data split across clients is moderately heterogeneous, we expect* Robust-HDP *to be successful in at least the decomposition task and the following noise estimation, given that the noise matrix* $\mathbf{S}$ *is sparse, and there are large enough number of clients.*

Whether the second condition is met or not, mainly depends on how much variation exists in the amount of noise in clients' model updates, i.e., how (approximately) sparse the set $\{\sigma_1^2, \cdots, \sigma_n^2\}$ is. As shown in Equations 3, 5 and 7, this mainly depends on clients' privacy parameters ($\{(\epsilon_i, \delta_i)\}_{i=1}^n$), and batch sizes ($\{b_i\}_{i=1}^n$), *and is independent of whether the data split is i.i.d or not.* The more the variation in clients' privacy parameters/batch sizes (similar to what we saw in Figure 2), the better we can consider $\mathbf{S}$ as an approximately sparse matrix, which validates our RPCA decomposition.

So far, we assumed an *i.i.d* data distribution across clients, which ensures that the underlying matrix $\mathbf{L}$ is indeed low-rank. Also, we assumed heterogeneity in batch size and privacy parameters of clients, which led to a sparse pattern in the noise matrix $\mathbf{S}$ (as shown in Figure 2, right). In order to evaluate Robust-HDP when the data split is moderately heterogeneous, we run experiments on MNIST with 60 clients in total (compared to the 20 clients before) and uniform batch size $b = 128$, and we split data such that each client holds data samples of at maximum 8 classes. The results obtained are reported in Table 4. As observed, Robust-HDP still outperforms the baselines in most of the cases. However, compared to the detailed results in Table 12, which were obtained for *i.i.d* data split, its superiority to the baseline algorithms has decreased. Detailed discussion of these results along with scenarios with highly heterogeneous data splits are reported in Appendix H.

5. Conclusion

In heterogeneous DPFL systems, heterogeneity in privacy preference, batch/dataset size results in large variations across the noise levels in clients' model updates, which existing algorithms can not fully take into account. To address this heterogeneity, we proposed a robust heterogeneous DPFL algorithm that performs noise-aware aggregation on an untrusted server, and is independent of clients' privacy parameter values shared with the server. The proposed algorithm is optimal, robust, vastly applicable, scalable, and improves utility and convergence speed.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Acknowledgements

We thank the reviewers and the area chair for the critical comments that have largely improved the final version of this paper. YY gratefully acknowledges funding support from NSERC, the Ontario early researcher program and the Canada CIFAR AI Chairs program. Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute. Also, YC acknowledges the support by JSPS KAKENHI JP22H03595, JST PRESTO JPMJPR23P5, JST CREST JPMJCR21M2.

References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016. URL https://doi.org/10.1145/2976749.2978318.

Alaggan, M., Gambs, S., and Kermarrec, A.-M. Heterogeneous differential privacy. *Journal of Privacy and Confidentiality*, 2017. URL https://journalprivacyconfidentiality.org/index.php/jpc/article/view/652.

Bagdasaryan, E. and Shmatikov, V. Differential privacy has disparate impact on model accuracy. In *Advances in Neural Information Processing Systems*, 2019. URL https://proceedings.neurips.cc/paper/2019/file/fc0de4e0396fff257ea362983c2dda5a-Paper.pdf.

Boenisch, F., Mühl, C., Dziedzic, A., Rinberg, R., and Papernot, N. Have it your way: Individualized privacy assignment for DP-SGD. In *Advances in Neural Information Processing Systems*, 2023. URL https://neurips.cc/virtual/2023/poster/71354.

Candes, E. J., Li, X., Ma, Y., and Wright, J. Robust principal component analysis?, 2009. URL https://arxiv.org/pdf/0912.3599.

Chathoth, A. K., Necciai, C. P., Jagannatha, A., and Lee, S. Differentially private federated continual learning with heterogeneous cohort privacy. In *2022 IEEE International Conference on Big Data (Big Data)*, 2022. URL https://ieeexplore.ieee.org/document/10021082.

Cummings, R., Gupta, V., Kimpara, D., and Morgenstern, J. On the compatibility of privacy and fairness. In *Adjunct Publication of the 27th Conference on User Modeling, Adaptation and Personalization*, 2019. URL https://doi.org/10.1145/3314183.3323847.

Deng, L. The MNIST database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 2012. URL https://ieeexplore.ieee.org/document/6296535.

Donahue, K. and Kleinberg, J. Model-sharing games: Analyzing federated learning under voluntary participation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021. URL https://ojs.aaai.org/index.php/AAAI/article/view/16669.

Dwork, C. A firm foundation for private data analysis. *Commun. ACM*, 2011. URL https://doi.org/10.1145/1866739.1866758.

Dwork, C. and Roth, A. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 2014. URL https://dl.acm.org/doi/10.1561/0400000042.

Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., and Naor, M. Our data, ourselves: Privacy via distributed noise generation. In *Proceedings of the 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2006a. URL https://doi.org/10.1007/11761679_29.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the Third Conference on Theory of Cryptography*. Springer-Verlag, 2006b. URL https://doi.org/10.1007/11681878_14.

Fallah, A., Makhdoumi, A., Malekian, A., and Ozdaglar, A. Optimal and differentially private data acquisition: Central and local mechanisms. *Operations Research*, 2023. URL https://arxiv.org/pdf/2201.03968.pdf.

Fioretto, F., Tran, C., Hentenryck, P. V., and Zhu, K. Differential privacy and fairness in decisions and learning tasks: A survey. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, 2022. URL https://doi.org/10.24963%2Fijcai.2022%2F766.

Geiping, J., Bauermeister, H., Dröge, H., and Moeller, M. Inverting gradients - how easy is it to break privacy in federated learning? *ArXiv*, 2020. URL https://proceedings.neurips.cc/paper/2020/file/c4ede56bbd98819ae6112b20ac6bf145-Paper.pdf.

Geyer, R. C., Klein, T., and Nabi, M. Differentially private federated learning: A client level perspective. *ArXiv*, 2017. URL https://arxiv.org/pdf/1712.07557.pdf.

Ghadimi, S. and Lan, G. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 2013. URL https://doi.org/10.1137/120880811.

Girgis, A. M., Data, D., Diggavi, S. N., Kairouz, P., and Suresh, A. T. Shuffled model of differential privacy in federated learning. In *AISTATS*, 2021. URL https://proceedings.mlr.press/v130/girgis21a.html.

Gur-Ari, G., Roberts, D. A., and Dyer, E. Gradient descent happens in a tiny subspace. *ArXiv*, 2018. URL https://arxiv.org/pdf/1812.04754.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. URL https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf.

Heo, G., Seo, J., and Whang, S. E. Personalized DP-SGD using sampling mechanisms. *ArXiv*, 2023. URL https://arxiv.org/pdf/2305.15165.

Hitaj, B., Ateniese, G., and Pérez-Cruz, F. Deep models under the GAN: Information leakage from collaborative deep learning. *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017. URL https://arxiv.org/pdf/1702.07464.

Huang, W., Zhou, S., Zhu, T., Liao, Y., Wu, C., and Qiu, S. Improving laplace mechanism of differential privacy by personalized sampling. In *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2020. URL https://ieeexplore.ieee.org/document/9343130.

Jorgensen, Z., Yu, T., and Cormode, G. Conservative or liberal? personalized differential privacy. In *2015 IEEE 31st International Conference on Data Engineering*, 2015. URL https://ieeexplore.ieee.org/document/7113353.

Kang, J., Pedarsani, R., and Ramchandran, K. The fair value of data under heterogeneous privacy constraints. *ArXiv*, 2023. URL https://arxiv.org/pdf/2301.13336.

Karimireddy, S. P., Guo, W., and Jordan, M. I. Mechanisms that incentivize data sharing in federated learning. *ArXiv*, 2022. URL https://arxiv.org/abs/2207.04557.

Kotsogiannis, I., Doudalis, S., Haney, S., Machanavajjhala, A., and Mehrotra, S. One-sided differential privacy. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, 2020. URL https://ieeexplore.ieee.org/document/9101725.

Krizhevsky, A. Learning multiple layers of features from tiny images, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Li, T., Hu, S., Beirami, A., and Smith, V. Ditto: Fair and robust federated learning through personalization. In *International Conference on Machine Learning*, 2020. URL http://proceedings.mlr.press/v139/li21h/li21h.pdf.

Liu, J., Lou, J., Xiong, L., Liu, J., and Meng, X. Projected federated averaging with heterogeneous differential privacy. *Proceedings of VLDB Endowment.*, 2021a. URL https://www.vldb.org/pvldb/vol15/p828-liu.pdf.

Liu, R., Cao, Y., Chen, H., Guo, R., and Yoshikawa, M. FLAME: Differentially private federated learning in the shuffle model. In *AAAI*, 2021b. URL https://www.aaai.org/AAAI21Papers/AAAI-4838.LiuR.pdf.

Malekmohammadi, S., Shaloudegi, K., Hu, Z., and Yu, Y. *A Unifying Framework for Federated Learning*, pp. 87–115. Springer International Publishing, Cham, 2023. ISBN 978-3-031-11748-0. doi: 10.1007/978-3-031-11748-0_5. URL https://doi.org/10.1007/978-3-031-11748-0_5.

Malekmohammadi, S., Taik, A., and Farnadi, G. Mitigating disparate impact of differential privacy in federated learning through robust clustering, 2024. URL https://arxiv.org/abs/2405.19272.

Matzken, C., Eger, S., and Habernal, I. Trade-Offs Between Fairness and Privacy in Language Modeling. In *Findings of the Association for Computational Linguistics: ACL*, 2023.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, 2017. URL http://proceedings.mlr.press/v54/mcmahan17a/mcmahan17a.pdf.

McMahan, H. B., Ramage, D., Talwar, K., and Zhang, L. Learning differentially private recurrent language models. In *ICLR*, 2018. URL https://arxiv.org/pdf/1710.06963.pdf.

Niu, B., Chen, Y., Wang, B., Cao, J., and Li, F. Utility-aware exponential mechanism for personalized differential privacy. In *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, 2020. URL https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9120532.

Noble, M., Bellet, A., and Dieuleveut, A. Differentially private federated learning on heterogeneous data. In *International Conference on Artificial Intelligence and Statistics*, 2021. URL https://proceedings.mlr.press/v151/noble22a/noble22a.pdf.

Rigaki, M. and García, S. A survey of privacy attacks in machine learning. *ArXiv*, 2020. URL https://arxiv.org/pdf/2007.07646.

Sattler, F., Müller, K.-R., and Samek, W. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE Transactions on Neural Networks and Learning Systems*, 2019. URL https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9174890.

Shi, W., Cui, A., Li, E., Jia, R., and Yu, Z. Selective differential privacy for language modeling. *ArXiv*, 2021. URL https://arxiv.org/pdf/2108.12944.pdf.

Wang, H., Yurochkin, M., Sun, Y., Papailiopoulos, D., and Khazaeni, Y. Federated learning with matched averaging. In *International Conference on Learning Representations*, 2019a. URL https://arxiv.org/pdf/2002.06440.

Wang, Z., Song, M., Zhang, Z., Song, Y., Wang, Q., and Qi, H. Beyond inferring class representatives: User-level privacy leakage from federated learning. *IEEE INFOCOM*, 2019b. URL https://arxiv.org/pdf/1812.00535.

Werner, M., He, L., Jordan, M., Jaggi, M., and Karimireddy, S. P. Provably personalized and robust federated learning. *Transactions on Machine Learning Research*, 2023. URL https://openreview.net/pdf?id=B0uBSSUy0G.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, 2017. URL http://arxiv.org/abs/1708.07747.

Yu, D., Kamath, G., Kulkarni, J., Liu, T.-Y., Yin, J., and Zhang, H. Individual privacy accounting for differentially private stochastic gradient descent. *ArXiv*, 2023. URL https://arxiv.org/abs/2206.02617.

Yu, L., Liu, L., Pu, C., Gursoy, M. E., and Truex, S. Differentially private model publishing for deep learning. *2019 IEEE Symposium on Security and Privacy (SP)*, 2019. URL https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8835283.

Zhang, G., Malekmohammadi, S., Chen, X., and Yu, Y. Proportional fairness in federated learning. *Transactions on Machine Learning Research*, 2023. URL https://openreview.net/forum?id=ryUHgEdWCQ.

Zhao, Y., Zhao, J., Yang, M., Wang, T., Wang, N., Lyu, L., Niyato, D., and Lam, K.-Y. Local differential privacy-based federated learning for internet of things. *IEEE Internet of Things Journal*, 2021. URL https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9253545.

Zhou, J., Su, Z., Ni, J., Wang, Y., Pan, Y., and Xing, R. Personalized privacy-preserving federated learning: Optimized trade-off between utility and privacy. In *GLOBECOM, IEEE Global Communications Conference*, 2022. URL https://ieeexplore.ieee.org/document/10000793.

Zhou, Y., Wu, S., and Banerjee, A. Bypassing the ambient dimension: Private SGD with gradient subspace identification. In *9th International Conference on Learning Representations, ICLR*, 2021. URL https://arxiv.org/pdf/2007.03813.

Zhu, L., Liu, Z., and Han, S. Deep leakage from gradients. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/60a6c4002cc7b29142def8871531281a-Paper.pdf.

Appendix for *Noise-Aware Algorithm for Heterogeneous differentially private federated learning*

A. Notations

We consider an FL setting with n clients. Let $x \in \mathcal{X} \subseteq \mathbb{R}^d$ and $y \in \mathcal{Y} = \{1, \dots, C\}$ denote an input data point and its target label. Client i holds dataset $\mathcal{D}_i = \{x_{ij}\}_{j=1}^{N_i}$ with N_i samples from distribution $P_i(x, y)$. Let $h : \mathcal{X} \times \boldsymbol{\theta} \to \mathbb{R}^C$ be the predictor function, which is parameterized by $\boldsymbol{\theta} \in \mathbb{R}^p$ (p is the number of model parameters) shared among all clients. Also, let $\ell : \mathbb{R}^C \times \mathcal{Y} \to \mathbb{R}_+$ be the loss function used (cross entropy loss). Following (McMahan et al., 2017), many existing FL algorithms fall into the natural formulation that minimizes the (arithmetic) average loss $f(\boldsymbol{\theta}) := \sum_{i=1}^{n} \lambda_i f_i(\boldsymbol{\theta})$, where $f_i(\boldsymbol{\theta}) = \frac{1}{N_i} \sum_{(x,y)\in\mathcal{D}_i} [\ell(h(x, \boldsymbol{\theta}), y)]$, with minimum value f_i^*. The weights $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_n)$ are nonnegative and sum to 1. At gradient update t, client i uses a data batch $\mathcal{B}_i^t$ with size $b_i = |\mathcal{B}_i^t|$. Let $q_i = \frac{b_i}{N_i}$ be batch size ratio of client i. There are E global communication rounds indexed by e, and in each of them, client i runs K_i local epochs. We use boldface letters to denote vectors.

B. Experimental setup

In this section, we provide more experimental details that were deferred to the appendix in the main paper.

B.1. Datasets and models

MNIST and FMNIST datasets: We consider a distributed setting with 20 clients. In order to create a heterogeneous dataset, we follow a similar procedure as in (McMahan et al., 2017): first we split the data from each class into several shards. Then, each user is randomly assigned a number of shards of data. For example, in some experiments, in order to guarantee that no user receives data from more than 8 classes, we split each class of MNIST/FMNIST into 16 shards (i.e., a total of 160 shards for the whole dataset), and each user is randomly assigned 8 shards of data. By considering 20 clients, this procedure guarantees that no user receives data from more than 8 classes and the data distribution of each user is different from each other. The local datasets are balanced–all clients have the same amount of training samples. In this way, each user has 2400 data points for training, and 600 for testing. We use a simple 2-layer CNN model with ReLU activation, the details of which can be found in Table 5. To update the local models at each user using its local data, unless otherwise is stated, we apply stochastic gradient descent (SGD).

Table 5. CNN model for classification on MNIST/FMNIST datasets

Layer	Output Shape	# of Trainable Parameters	Activation	Hyper-parameters
Input	$(1, 28, 28)$	0		
Conv2d	$(16, 28, 28)$	416	ReLU	kernel size =5; strides=$(1, 1)$
MaxPool2d	$(16, 14, 14)$	0		pool size=$(2, 2)$
Conv2d	$(32, 14, 14)$	12,832	ReLU	kernel size =5; strides=$(1, 1)$
MaxPool2d	$(32, 7, 7)$	0		pool size=$(2, 2)$
Flatten	1568	0		
Dense	10	15,690	ReLU	
Total		28,938		

CIFAR10/100 datasets: We consider a distributed setting with 20 clients, and split the 50,000 training samples and the 10,000 test samples in the datasets among them. In order to create a dataset, we follow a similar procedure as in (McMahan et al., 2017): For instance for CIFAR10, first we sort all data points according to their classes. Then, each class is split into 20 shards, and each user is randomly assigned 1 shard of each class. We use the residual neural network (ResNet-18) defined in (He et al., 2015), which is a large model with $p = 11,181,642$ parameters for CIFAR10. We also use ResNet-34 (He et al., 2015), which is a larger model with $p = 21,272,778$ parameters for CIFAR100. To update the local models at each user using its local data, we apply stochastic gradient descent (SGD). In the reported experimental results, all clients participate in each communication round.

Table 6. Details of the experiments and the used datasets in the main body of the paper. ResNet-18/34 are the residual neural networks defined in (He et al., 2015). CNN: Convolutional Neural Network defined in Table 5.

Datasets	Train set size	Test set size	Data Partition method	# of clients	Model	# of parameters
MNIST	48000	12000	sharding (McMahan et al., 2017)	20/40/60	CNN (Table 5)	28,938
FMNIST	50000	10000	sharding (McMahan et al., 2017)	20	CNN (Table 5)	28,938
CIFAR10	50000	10000	sharding (McMahan et al., 2017)	20	ResNet-18 (He et al., 2015)	11,181,642
CIFAR100	50000	10000	sharding (McMahan et al., 2017)	20	ResNet-34 (He et al., 2015)	21,272,778

Distribution	Parameter setting
Dist1	Gaussian distribution $\mathcal{N}(2.0, 1.0)$
Dist2	mixture of $\mathcal{N}(0.2, 0.01)$, $\mathcal{N}(1.0, 0.1)$ and $\mathcal{N}(5.0, 1.0)$ with weights $(0.2, 0.6, 0.2)$
Dist3	Uniform distribution $U[0.2, 5]$
Dist4	mixture of $\mathcal{N}(0.2, 0.01)$, $\mathcal{N}(0.5, 0.1)$ and $\mathcal{N}(2.0, 1.0)$ with weights $(0.2, 0.6, 0.2)$
Dist5	Uniform distribution $U[0.2, 2]$
Dist6	mixture of $\mathcal{N}(0.2, 0.01)$, $\mathcal{N}(0.5, 0.1)$ and $\mathcal{N}(1.0, 0.1)$ with weights $(0.3, 0.5, 0.2)$
Dist7	Uniform distribution $U[0.2, 1]$
Dist8	mixture of $\mathcal{N}(0.2, 0.01)$ and $\mathcal{N}(0.5, 0.1)$ with weights $(0.6, 0.4)$
Dist9	Uniform distribution $U[0.2, 0.5]$

Table 7. Distributions of privacy parameters (ϵ), from which we sample clients' privacy parameters.

B.2. DP training parameters

For each dataset, we sample the privacy parameter ϵ of clients from different distributions, as shown in Table 7. In order to get reasonable accuracy results for CIFAR100, which is a harder dataset compared to the other three datasets, we scale the values of ϵ sampled for clients from the distributions above by a factor 10. For instance, we have $\mathcal{N}(20.0, 10.0)$ as "Dist1" for CIFAR100. This is only for getting meaningful accuracy values for CIFAR100, otherwise the test accuracy values will be too low. We fix δ for all clients to 10^{-4}. We also set the clipping threshold c equal to 3, as it results in better test accuracy, as reported in (Abadi et al., 2016).

B.3. Algorithms to compare and tuning hyperparameters

We compare our Robust-HDP, which benefits from RPCA (Algorithm 3), with four baseline algorithms, including WeiAvg (Liu et al., 2021a) (Algorithm 2), PFA (Liu et al., 2021a), DPFedAvg (Noble et al., 2021) and minimum ϵ (Liu et al., 2021a). For PFA, we always use projection dimension 1, as in (Liu et al., 2021a). For each algorithm and each dataset, we find the best learning rate from a grid: *the one which is small enough to avoid divergence of the federated optimization, and results in the lowest average train loss (across clients) at the end of FL training*. Here are the grids we use for each dataset:

- MNIST: {1e-4, 2e-4, 5e-4, 1e-3, 2e-3, 5e-3, 1e-2};
- FMNIST: {1e-4, 2e-4, 5e-4, 1e-3, 2e-3, 5e-3, 1e-2};
- CIFAR10: {1e-4, 2e-4, 5e-4, 1e-3, 2e-3, 5e-3, 1e-2};
- CIFAR100: {1e-5, 2e-5, 5e-5, 1e-4, 2e-4, 5e-4, 1e-3}.

The best learning rates used for each dataset are reported in Table 8 to Table 11.

Algorithm 2 WeiAvg (Liu et al., 2021a)

```
Input: Initial parameter $\boldsymbol{\theta}^0$, Clients batch sizes $\{b_1, \dots, b_n\}$, Clients dataset sizes $\{N_1, \dots, N_n\}$,
Clients noise scales $\{z_1, \dots, z_n\}$, gradient norm bound $c$, local epochs $\{K_1, \dots, K_n\}$, global round $E$,
privacy parameter $\delta$, number of model parameters $p$, privacy accountant PA.
Output: $\boldsymbol{\theta}_E, \{\epsilon_1, \dots, \epsilon_n\}$
Initialize $\boldsymbol{\theta}_0$ randomly.
for $e \in [E]$ do
    sample a set of clients $\mathcal{S}^e \subseteq \{1, \dots, n\}$
    for each client $i \in \mathcal{S}^e$ in parallel do
        $\Delta\tilde{\boldsymbol{\theta}}_i^e \leftarrow$ DPSGD $(\boldsymbol{\theta}^e, b_i, N_i, K_i, z_i, c)$
        $\epsilon_i^e \leftarrow \mathbf{PA}(\frac{b_i}{N_i}, z_i, K_i, e)$
    for $i \in \mathcal{S}^e$ do
        $w_i^e \leftarrow \frac{\epsilon_i}{\sum_{j \in \mathcal{S}_e} \epsilon_j}$
    $\boldsymbol{\theta}^{e+1} \leftarrow \boldsymbol{\theta}^e + \sum_{i \in \mathcal{S}_e} w_i^e \Delta\tilde{\boldsymbol{\theta}}_i^e$
Output: $\boldsymbol{\theta}^E, \{\epsilon_1^E, \dots, \epsilon_n^E\}$
```

Algorithm 3 Principal Component Pursuit by Alternating Directions (Candes et al., 2009)

```
Input: matrix $M$, shrinkage operator $\mathcal{S}_\tau[x] = \text{sgn}(x)\max(|x| - \tau, 0)$, singular value thresholding operator
$\mathcal{D}_\tau(U\Sigma V^*) = U\mathcal{S}_\tau(\Sigma)V^*$
Initialize $S_0 = Y_0 = 0, \mu > 0$.
while not converged do
    compute $L_{k+1} = \mathcal{D}_{\mu^{-1}}(M - S_k - \mu^{-1}Y_k)$
    compute $S_{k+1} = \mathcal{S}_{\lambda\mu^{-1}}(M - L_{k+1} + \mu^{-1}Y_k)$
    compute $Y_{k+1} = Y_k + \mu(M - L_{k+1} - S_{k+1})$
Output: $L, S$
```

Table 8. The learning rates used for training with each algorithm on MNIST dataset

dist alg	Dist1	Dist2	Dist3	Dist4	Dist5	Dist6	Dist7	Dist8	Dist9
WeiAvg (Liu et al., 2021a)	1e-2	5e-3	1e-2	5e-3	5e-3	1e-3	1e-3	1e-3	1e-3
PFA (Liu et al., 2021a)	5e-3	5e-3	5e-3	5e-3	5e-3	1e-3	1e-3	1e-3	5e-4
DPFedAvg (Noble et al., 2021)	5e-3	1e-3	1e-3	1e-3	1e-3	5e-4	1e-3	1e-3	1e-3
minimum ϵ (Liu et al., 2021a)	5e-4	5e-4	5e-4	5e-4	1e-3	1e-4	1e-3	5e-4	1e-3
Robust-HDP	1e-2	1e-2	1e-2	1e-2	5e-3	2e-3	2e-3	2e-3	2e-3

Table 9. The learning rates used for training with each algorithm on FMNIST dataset

dist alg	Dist1	Dist2	Dist3	Dist4	Dist5	Dist6	Dist7	Dist8	Dist9
WeiAvg (Liu et al., 2021a)	5e-3	5e-3	5e-3	5e-3	2e-3	5e-4	5e-4	5e-4	5e-4
PFA (Liu et al., 2021a)	2e-3	2e-3	5e-3	5e-3	5e-3	5e-3	2e-3	1e-3	1e-3
DPFedAvg (Noble et al., 2021)	2e-3	1e-3	1e-3	1e-3	1e-3	5e-4	5e-4	5e-4	5e-4
minimum ϵ (Liu et al., 2021a)	1e-3	5e-4	5e-4	5e-4	5e-4	1e-4	5e-4	5e-4	5e-4
Robust-HDP	5e-3	5e-3	5e-3	5e-3	5e-3	1e-3	1e-3	1e-3	1e-3

Table 10. The learning rates used for training with each algorithm on CIFAR10 dataset

dist alg	Dist1	Dist2	Dist3	Dist4	Dist5	Dist6	Dist7	Dist8	Dist9
WeiAvg (Liu et al., 2021a)	2e-3	1e-3	1e-3	5e-4	5e-4	2e-4	2e-4	2e-4	2e-4
PFA (Liu et al., 2021a)	2e-3	2e-3	2e-3	2e-3	2e-3	1e-3	5e-4	5e-4	2e-4
DPFedAvg (Noble et al., 2021)	1e-3	5e-4	2e-4	2e-4	2e-4	1e-4	1e-4	5e-5	1e-4
minimum ϵ (Liu et al., 2021a)	2e-3	1e-3	1e-3	1e-3	1e-3	1e-4	5e-4	2e-4	2e-4
Robust-HDP	2e-3	2e-3	2e-3	2e-3	2e-3	5e-4	1e-3	2e-4	2e-4

Table 11. The learning rates used for training with each algorithm on CIFAR100 dataset.

dist alg	Dist1	Dist2	Dist3	Dist4	Dist5	Dist6	Dist7	Dist8	Dist9
WeiAvg (Liu et al., 2021a)	1e-3	1e-3	1e-3	5e-4	5e-4	2e-4	2e-4	2e-4	2e-4
PFA (Liu et al., 2021a)	2e-3	2e-3	2e-3	1e-3	1e-3	5e-4	2e-4	2e-4	1e-4
DPFedAvg (Noble et al., 2021)	5e-4	5e-4	1e-4	1e-4	1e-4	5e-5	5e-5	2e-5	2e-5
minimum ϵ (Liu et al., 2021a)	2e-4	2e-4	1e-4	1e-4	1e-4	5e-5	5e-5	2e-5	2e-5
Robust-HDP	2e-3	2e-3	2e-3	2e-3	2e-3	1e-3	1e-3	1e-3	1e-3

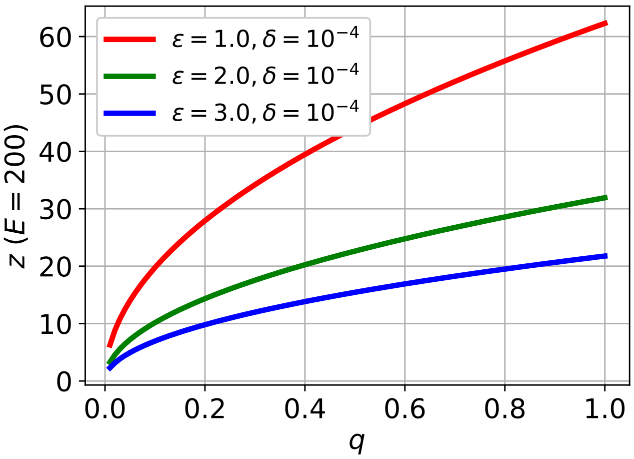

Figure 8. Plot of z v.s. q obtained from Moments Accountant (Abadi et al., 2016) in a centralized setting with $E = 200$. As observed, z increases sub-linearly with q (or equivalently with b) and decreases with dataset size.

C. Derivations

Computation of $\sigma^2_{i,\tilde{g}}$, when gradient clipping is effective for all samples: We know that the two sources of randomness (i.e., minibatch sampling and Gaussian noise) are independent, thus the variance is additive. Assuming that $E[\bar{g}_{ij}(\boldsymbol{\theta})]$ is the same for all j and is $G_i(\boldsymbol{\theta})$, we have:

$$\begin{aligned}
\sigma^2_{i,\tilde{g}} := \texttt{Var}(\tilde{g}_i(\boldsymbol{\theta})) &= \texttt{Var}\left(\frac{1}{b_i}\sum_{j\in\mathcal{B}_i^t}\bar{g}_{ij}(\boldsymbol{\theta})\right) + \frac{p\sigma^2_{i,\texttt{DP}}}{b_i^2} \\
&= \frac{1}{b_i^2}\left(\mathbb{E}\left[\left\|\sum_{j\in\mathcal{B}_i^t}\bar{g}_{ij}(\boldsymbol{\theta})\right\|^2\right] - \left\|\mathbb{E}\left[\sum_{j\in\mathcal{B}_i^t}\bar{g}_{ij}(\boldsymbol{\theta})\right]\right\|^2\right) + \frac{pc^2z^2(\epsilon_i,\delta_i,q_i,K_i,E)}{b_i^2} \\
&= \frac{1}{b_i^2}\left(\mathbb{E}\left[\left\|\sum_{j\in\mathcal{B}_i^t}\bar{g}_{ij}(\boldsymbol{\theta})\right\|^2\right] - \left\|\sum_{j\in\mathcal{B}_i^t}G_i(\boldsymbol{\theta})\right\|^2\right) + \frac{pc^2z^2(\epsilon_i,\delta_i,q_i,K_i,E)}{b_i^2} \\
&= \frac{1}{b_i^2}\left(\mathbb{E}\left[\left\|\sum_{j\in\mathcal{B}_i^t}\bar{g}_{ij}(\boldsymbol{\theta})\right\|^2\right] - b_i^2\left\|G_i(\boldsymbol{\theta})\right\|^2\right) + \frac{pc^2z^2(\epsilon_i,\delta_i,q_i,K_i,E)}{b_i^2} \qquad (11)
\end{aligned}$$

We also have:

$$\begin{aligned}
\mathbb{E}\left[\left\|\sum_{j\in\mathcal{B}_i^t}\bar{g}_{ij}(\boldsymbol{\theta})\right\|^2\right] &= \sum_{j\in\mathcal{B}_i^t}\mathbb{E}\left[\left\|\bar{g}_{ij}(\boldsymbol{\theta})\right\|^2\right] + \sum_{m\neq n\in\mathcal{B}_i^t}2\mathbb{E}\left[[\bar{g}_{im}(\boldsymbol{\theta})]^\top[\bar{g}_{in}(\boldsymbol{\theta})]\right] \\
&= \sum_{j\in\mathcal{B}_i^t}\mathbb{E}\left[\left\|\bar{g}_{ij}(\boldsymbol{\theta})\right\|^2\right] + \sum_{m\neq n\in\mathcal{B}_i^t}2\mathbb{E}\left[\bar{g}_{im}(\boldsymbol{\theta})\right]^\top\mathbb{E}\left[\bar{g}_{in}(\boldsymbol{\theta})\right] \\
&= b_ic^2 + 2\binom{b_i}{2}\left\|G_i(\boldsymbol{\theta})\right\|^2, \qquad (12)
\end{aligned}$$

where the last equation has used Equation (2) and that we clip the norm of sample gradients $\bar{g}_{ij}(\boldsymbol{\theta})$ with an "effective" clipping threshold c. We can now plug eq. 12 into the parenthesis in eq. 11 and rewrite it as:

$$\begin{aligned}
\sigma^2_{i,\tilde{g}} := \texttt{Var}(\tilde{g}_i(\boldsymbol{\theta})) &= \frac{1}{b_i^2}\left(\mathbb{E}\left[\left\|\sum_{j\in\mathcal{B}_i^t}\bar{g}_{ij}(\boldsymbol{\theta})\right\|^2\right] - b_i^2\left\|G_i(\boldsymbol{\theta})\right\|^2\right) + \frac{pc^2z^2(\epsilon_i,\delta_i,q_i,K_i,E)}{b_i^2} \\
&= \frac{1}{b_i^2}\left(b_ic^2 + \left(2\binom{b_i}{2} - b_i^2\right)\left\|G_i(\boldsymbol{\theta})\right\|^2\right) + \frac{pc^2z^2(\epsilon_i,\delta_i,q_i,K_i,E)}{b_i^2} \\
&= \frac{1}{b_i^2}\left(b_ic^2 - b_i\left\|G_i(\boldsymbol{\theta})\right\|^2\right) + \frac{pc^2z^2(\epsilon_i,\delta_i,q_i,K_i,E)}{b_i^2} \\
&= \frac{c^2 - \left\|G_i(\boldsymbol{\theta})\right\|^2}{b_i} + \frac{pc^2z^2(\epsilon_i,\delta_i,q_i,K_i,E)}{b_i^2} \\
&\approx \frac{pc^2z^2(\epsilon_i,\delta_i,q_i,K_i,E)}{b_i^2} \qquad (13)
\end{aligned}$$

D. Assumptions and lemmas

In this section, we formalize our assumptions and some lemmas, which we will use in our proofs.

Assumption D.1 (Lipschitz continuity, β-smoothness and bounded gradient variance). $\{f_i\}_{i=1}^n$ are L_0-Lipschitz continuous and β-smooth: $\forall\ \boldsymbol{\theta}, \boldsymbol{\theta}' \in \mathbb{R}^p, i : \|f_i(\boldsymbol{\theta}) - f_i(\boldsymbol{\theta}')\| \leq L_0\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|$ and $\|\nabla f_i(\boldsymbol{\theta}) - \nabla f_i(\boldsymbol{\theta}')\| \leq \beta\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|$. Also, the stochastic gradient $g_i(\boldsymbol{\theta})$ is an unbiased estimate of $\nabla f_i(\boldsymbol{\theta})$ with bounded variance: $\forall \boldsymbol{\theta} \in \mathbb{R}^p : \mathbb{E}_{\mathcal{B}_i^t}[g_i(\boldsymbol{\theta})] = \nabla f_i(\boldsymbol{\theta}),\ \ \mathbb{E}_{\mathcal{B}_i^t}\left[\|g_i(\boldsymbol{\theta}) - \nabla f_i(\boldsymbol{\theta})\|^2\right] \leq \sigma_{i,g}^2$. We also assume that for every $i, j \in [n]$, $f_i - f_j$ is σ-Lipschitz continuous: $\|\nabla f_i(\boldsymbol{\theta}) - \nabla f_j(\boldsymbol{\theta})\| \leq \sigma$.

Assumption D.2 (bounded sample gradients). There exists a clipping threshold $\mathcal{C}$ such that for all i, j:

$$\|g_{ij}(\boldsymbol{\theta})\|_2 := \|\nabla \ell(h(x_{ij}, \boldsymbol{\theta}), y_{ij})\|_2 \leq \mathcal{C} \tag{14}$$

Note that this condition always holds if ℓ is Lipschitz continuous or if h is bounded.

Lemma D.3 (Relaxed triangle inequality). *Let $\{v_1, \ldots, v_n\}$ be n vectors in $\mathbb{R}^d$. Then, the followings is true:*

- $\|v_i + v_j\|^2 \leq (1+a)\|v_i\|^2 + (1 + \frac{1}{a})\|v_j\|^2$ *(for any $a > 0$)*
- $\|\sum_i v_i\|^2 \leq n \sum_i \|v_i\|^2$

Proof. The proof for the first inequality is obtained from identity:

$$\|v_i + v_j\|^2 = (1+a)\|v_i\|^2 + (1 + \frac{1}{a})\|v_j\|^2 - \|\sqrt{a}v_i + \frac{1}{\sqrt{a}}v_j\|^2 \tag{15}$$

The proof for the second inequality is achieved by using the fact that $h(x) = \|x\|^2$ is convex:

$$\|\frac{1}{n}\sum_i v_i\|^2 \leq \frac{1}{n}\sum_i \|v_i\|^2 \tag{16}$$

□

Lemma D.4. *Let $\{v_1, \ldots, v_n\}$ be n random variables in $\mathbb{R}^d$, with $\mathbb{E}[v_i] = \mathcal{E}_i$ and $\mathbb{E}[\|v_i - \mathcal{E}_i\|^2] = \sigma_i^2$. Then, we have the following inequality:*

$$\mathbb{E}[\|\sum_{i=1}^n v_i\|^2] \leq \|\sum_{i=1}^n \mathcal{E}_i\|^2 + n\sum_{i=1}^n \sigma_i^2. \tag{17}$$

Proof. From the definition of variance, we have:

$$\mathbb{E}[\|\sum_{i=1}^n v_i\|^2] = \|\sum_{i=1}^n \mathcal{E}_i\|^2 + \mathbb{E}[\|\sum_{i=1}^n (v_i - \mathcal{E}_i)\|^2] \tag{18}$$

$$\leq \|\sum_{i=1}^n \mathcal{E}_i\|^2 + n\sum_{i=1}^n \mathbb{E}[\|v_i - \mathcal{E}_i\|^2] \tag{19}$$

$$= \|\sum_{i=1}^n \mathcal{E}_i\|^2 + n\sum_{i=1}^n \sigma_i^2, \tag{20}$$

$$\tag{21}$$

where the inequality is based on the Lemma D.3. □

Property D.5 (Parallel Composition (Yu et al., 2019)). *Assume each of the randomized mechanisms $M_i : \mathcal{D}_i \to \mathbb{R}$ for $i \in [n]$ satisfies (ϵ_i, δ_i)-$\texttt{DP}$ and their domains $\mathcal{D}_i$ are disjoint subsets. Any function g of the form $g(M_1, \ldots, M_n)$ satisfies $(\max_i \epsilon_i, \max_i \delta_i)$-$\texttt{DP}$.*

E. Proofs

Theorem 3.1. *For each client i , there exist constants c_1 and c_2 such that given its number of steps $E \cdot E_i$, for any $\epsilon < c_1 q_i^2 E \cdot E_i$, the output model of* Robust-HDP *satisfies $(\epsilon_i, \delta_i)-\texttt{DP}$ with respect to $\mathcal{D}_i$ for any $\delta_i > 0$ if $z_i > c_2 \frac{q_i \sqrt{E \cdot E_i \cdot \log \frac{1}{\delta_i}}}{\epsilon_i}$, where z_i is the noise scale used by the client i for* DPSGD. *The algorithm also satisfies $(\epsilon_{\texttt{max}}, \delta_{\texttt{max}})$-$\texttt{DP}$, where $(\epsilon_{\texttt{max}}, \delta_{\texttt{max}}) = \big(\max(\{\epsilon_i\}_{i=1}^n), \max(\{\delta_i\}_{i=1}^n)\big)$.*

Proof. The proof for the first part follows the proof of DPSGD algorithm (Abadi et al., 2016). Also, in Robust-HDP, each client i runs DPSGD locally to achieve (ϵ_i, δ_i)-$\texttt{DP}$ independently. Hence, it satisfies heterogeneous $\texttt{DP}$ with the set of preferences $\{(\epsilon_i, \delta_i)\}_{i=1}^n$. Also, the clients datasets $\{\mathcal{D}_i\}_{i=1}^n$ are disjoint. Hence, as Robust-HDP runs RPCA on the clients models updates, it satisfies $\big(\max(\{\epsilon_i\}_{i=1}^n), \max(\{\delta_i\}_{i=1}^n)\big)$-$\texttt{DP}$, according to parallel composition property above. □

Theorem 3.2 (Robust-HDP). *Assume that Assumptions D.1 and D.2 hold, and for every i, learning rate η_l satisfies: $\eta_l \leq \frac{1}{6\beta E_i}$ and $\eta_l \leq \frac{1}{12\beta\sqrt{(1+\sum_{i=1}^n E_i)\big(\sum_{i=1}^n E_i^4\big)}}$. Then, we have:*

$$\min_{0 \leq e \leq E-1} \mathbb{E}[\|\nabla f(\boldsymbol{\theta}^e)\|^2] \\ \leq \frac{12}{11E_l^{\texttt{min}} - 7}\bigg(\frac{f(\boldsymbol{\theta}^0) - f^*}{E\eta_l} + \Psi_\sigma + \Psi_p\bigg), \tag{10}$$

where $E_l^{\texttt{min}} = \min_i E_i$, i.e., the minimum number of local SGD steps across clients. Also, Ψ_p and Ψ_σ are two constants controlling the quality of the final model parameter returned by Robust-HDP*, which are explained in the following.*

Proof. From our assumption D.1 and that we use cross-entropy loss, we can conclude that Assumption D.2 also holds for some $\mathcal{C}$. In that case, we have:

$$\tilde{g}_i(\boldsymbol{\theta}) = \frac{\sum_{j \in \mathcal{B}_i^t} g_{ij}(\boldsymbol{\theta})}{b_i} + \mathcal{N}(\mathbf{0}, \frac{\sigma_{i,\texttt{DP}}^2}{b_i^2}\mathbb{I}_p) = g_i(\boldsymbol{\theta}) + \mathcal{N}(\mathbf{0}, \frac{\sigma_{i,\texttt{DP}}^2}{b_i^2}\mathbb{I}_p) \tag{22}$$

Therefore:

$$\mathbb{E}[\tilde{g}_i(\boldsymbol{\theta})] = \mathbb{E}[g_i(\boldsymbol{\theta})] = \nabla f_i(\boldsymbol{\theta}) \\ \texttt{Var}(\tilde{g}_i(\boldsymbol{\theta})) = \texttt{Var}(g_i(\boldsymbol{\theta})) + \frac{p\sigma_{i,\texttt{DP}}^2}{b_i^2} \leq \sigma_{i,\tilde{g}}^2 := \sigma_{i,g}^2 + \frac{p\sigma_{i,\texttt{DP}}^2}{b_i^2}. \tag{23}$$

i.e., the assumption of having unbiased gradient with bounded variance still holds (with a larger bound $\sigma_{i,\tilde{g}}^2$, due to adding $\texttt{DP}$ noise). Consistent with the previous notations, we assume that the set of participating clients in round e are $\mathcal{S}^e$, and for every client $i \notin \mathcal{S}^e$, we set $w_i^e = 0$. Using this, we can write the model parameter at the end of round e as:

$$\boldsymbol{\theta}^{e+1} = \sum_{i=1}^n w_i^e \boldsymbol{\theta}_{i,E_i}^e, \tag{24}$$

where $\{E_i\}_{i=1}^n$ is the heterogeneous number of gradient steps of clients (depending on their dataset size and batch size). From $\boldsymbol{\theta}_{i,k}^e = \boldsymbol{\theta}_{i,k-1}^e - \eta_l \tilde{g}_i(\boldsymbol{\theta}_{i,k-1}^e)$, we can rewrite the equation above as:

$$\boldsymbol{\theta}^{e+1} = \boldsymbol{\theta}^e - \eta_l \sum_{i \in \mathcal{S}^e} w_i^e \sum_{k=1}^{E_i} \tilde{g}_i(\boldsymbol{\theta}_{i,k-1}^e) = \boldsymbol{\theta}^e - \eta_l \sum_{i=1}^n w_i^e \sum_{k=1}^{E_i} \tilde{g}_i(\boldsymbol{\theta}_{i,k-1}^e) = \boldsymbol{\theta}^e - \eta_l \sum_{i=1}^n w_i^e \sum_{k=0}^{E_i-1} \tilde{g}_i(\boldsymbol{\theta}_{i,k}^e) \tag{25}$$

Note that the second equality holds because we assumed above that if client i is not participating in round e (i.e., $i \notin \mathcal{S}^e$), we set $w_i^e = 0$. From β-smoothness of $\{f_i\}_{i=1}^n$, and consequently β-smoothness of f, we have:

$$\begin{aligned} f(\boldsymbol{\theta}^{e+1}) &\leq f(\boldsymbol{\theta}^{e}) + \langle \nabla f(\boldsymbol{\theta}^{e}), \boldsymbol{\theta}^{e+1} - \boldsymbol{\theta}^{e} \rangle + \frac{\beta}{2} \| \boldsymbol{\theta}^{e+1} - \boldsymbol{\theta}^{e} \|^2 \\ &= f(\boldsymbol{\theta}^{e}) - \eta_l \big\langle \nabla f(\boldsymbol{\theta}^{e}), \sum_{i=1}^{n} w_i^e \sum_{k=0}^{E_i-1} \tilde{g}_i(\boldsymbol{\theta}^{e}_{i,k}) \big\rangle + \frac{\beta \eta_l^2}{2} \big\| \sum_{i=1}^{n} w_i^e \sum_{k=0}^{E_i-1} \tilde{g}_i(\boldsymbol{\theta}^{e}_{i,k}) \big\|^2 \end{aligned} \tag{26}$$

Now, we use identity $\tilde{g}_i(\boldsymbol{\theta}^{e}_{i,k}) = \nabla f(\boldsymbol{\theta}^{e}) + \tilde{g}_i(\boldsymbol{\theta}^{e}_{i,k}) - \nabla f(\boldsymbol{\theta}^{e})$ to rewrite the equation above as:

$$\begin{aligned} f(\boldsymbol{\theta}^{e+1}) \leq f(\boldsymbol{\theta}^{e}) &- \eta_l \big\langle \nabla f(\boldsymbol{\theta}^{e}), \sum_{i=1}^{n} w_i^e \sum_{k=0}^{E_i-1} \nabla f(\boldsymbol{\theta}^{e}) \big\rangle - \eta_l \big\langle \nabla f(\boldsymbol{\theta}^{e}), \sum_{i=1}^{n} w_i^e \sum_{k=0}^{E_i-1} \big(\tilde{g}_i(\boldsymbol{\theta}^{e}_{i,k}) - \nabla f(\boldsymbol{\theta}^{e}) \big) \big\rangle \\ &+ \frac{\beta \eta_l^2}{2} \big\| \sum_{i=1}^{n} w_i^e \sum_{k=0}^{E_i-1} \big(\tilde{g}_i(\boldsymbol{\theta}^{e}_{i,k}) - \nabla f(\boldsymbol{\theta}^{e}) \big) + \sum_{i=1}^{n} w_i^e E_i \nabla f(\boldsymbol{\theta}^{e}) \big\|^2 \end{aligned} \tag{27}$$

Hence,

$$\begin{aligned} f(\boldsymbol{\theta}^{e+1}) \leq f(\boldsymbol{\theta}^{e}) &- \eta_l \big\langle \nabla f(\boldsymbol{\theta}^{e}), \sum_{i=1}^{n} w_i^e \sum_{k=0}^{E_i-1} \nabla f(\boldsymbol{\theta}^{e}) \big\rangle - \eta_l \big\langle \nabla f(\boldsymbol{\theta}^{e}), \sum_{i=1}^{n} w_i^e \sum_{k=0}^{E_i-1} \big(\tilde{g}_i(\boldsymbol{\theta}^{e}_{i,k}) - \nabla f(\boldsymbol{\theta}^{e}) \big) \big\rangle \\ &+ \frac{\beta \eta_l^2}{2} \big\| \sum_{i=1}^{n} w_i^e \sum_{k=0}^{E_i-1} \big(\tilde{g}_i(\boldsymbol{\theta}^{e}_{i,k}) - \nabla f(\boldsymbol{\theta}^{e}) \big) \|^2 + \frac{\beta \eta_l^2}{2} \underbrace{(\sum_{i=1}^{n} w_i^e E_i)^2}_{\bar{E}_l^{e\,2}} \| \nabla f(\boldsymbol{\theta}^{e}) \big\|^2 \\ &+ \beta \eta_l^2 \big\langle \sum_{i=1}^{n} w_i^e E_i \nabla f(\boldsymbol{\theta}^{e}), \sum_{i=1}^{n} w_i^e \sum_{k=0}^{E_i-1} \big(\tilde{g}_i(\boldsymbol{\theta}^{e}_{i,k}) - \nabla f(\boldsymbol{\theta}^{e}) \big) \big\rangle. \end{aligned} \tag{28}$$

Note that we denote $\sum_{i=1}^{n} w_i^e E_i$ with $\bar{E}_l^e$ from now on. With doing some algebra we get to:

$$\begin{aligned} f(\boldsymbol{\theta}^{e+1}) &\leq f(\boldsymbol{\theta}^{e}) - \eta_l \bar{E}_l^e (1 - \frac{\beta}{2} \eta_l \bar{E}_l^e) \| \nabla f(\boldsymbol{\theta}^{e}) \|^2 \\ &- \eta_l (1 - \beta \eta_l \bar{E}_l^e) \big\langle \nabla f(\boldsymbol{\theta}^{e}), \sum_{i=1}^{n} w_i^e \sum_{k=0}^{E_i-1} \big(\tilde{g}_i(\boldsymbol{\theta}^{e}_{i,k}) - \nabla f(\boldsymbol{\theta}^{e}) \big) \big\rangle \\ &+ \frac{\beta \eta_l^2}{2} \Big\| \sum_{i=1}^{n} w_i^e \sum_{k=0}^{E_i-1} \big(\tilde{g}_i(\boldsymbol{\theta}^{e}_{i,k}) - \nabla f(\boldsymbol{\theta}^{e}) \big) \Big\|^2. \end{aligned} \tag{29}$$

By taking expectation from both side (expectation is conditioned on $\boldsymbol{\theta}^{e}$) and using Cauchy-Schwarz inequality, we have:

$$\begin{aligned} \mathbb{E}\big[f(\boldsymbol{\theta}^{e+1})\big] &\leq \mathbb{E}\big[f(\boldsymbol{\theta}^{e})\big] - \eta_l \bar{E}_l^e (1 - \frac{\beta \eta_l}{2} \bar{E}_l^e) \mathbb{E}\big[\| \nabla f(\boldsymbol{\theta}^{e}) \|^2\big] \\ &+ \eta_l (1 - \beta \eta_l \bar{E}_l^e) \mathbb{E}\Big[\| \nabla f(\boldsymbol{\theta}^{e}) \| \times \Big\| \sum_{i=1}^{n} w_i^e \sum_{k=0}^{E_i-1} \big(\nabla f_i(\boldsymbol{\theta}^{e}_{i,k}) - \nabla f(\boldsymbol{\theta}^{e}) \big) \Big\| \Big] \\ &+ \frac{\beta \eta_l^2}{2} \mathbb{E}\Big[\Big\| \sum_{i=1}^{n} w_i^e \sum_{k=0}^{E_i-1} \big(\tilde{g}_i(\boldsymbol{\theta}^{e}_{i,k}) - \nabla f(\boldsymbol{\theta}^{e}) \big) \Big\|^2 \Big]. \end{aligned} \tag{30}$$

Now, we use the inequality $ab \leq \frac{1}{2}(a^2+b^2)$ for the second line to get:

$$\begin{aligned}\mathbb{E}\big[f(\boldsymbol{\theta}^{e+1})\big] \leq \mathbb{E}\big[f(\boldsymbol{\theta}^{e})\big] &+ \underbrace{\big(\frac{1}{2}\eta_l(1-\beta\eta_l\bar{E}_l^e) - \eta_l\bar{E}_l^e(1-\frac{\beta\eta_l}{2}\bar{E}_l^e)\big)}_{\leq -\eta_l\frac{11\bar{E}_l^e-6}{12}}\mathbb{E}\big[\|\nabla f(\boldsymbol{\theta}^e)\|^2\big] \\ &+ \frac{1}{2}\eta_l(1-\beta\eta_l\bar{E}_l^e)\mathbb{E}\Big[\Big\|\sum_{i=1}^{n} w_i^e \sum_{k=0}^{E_i-1}\big(\nabla f_i(\boldsymbol{\theta}_{i,k}^e) - \nabla f(\boldsymbol{\theta}^e)\big)\Big\|^2\Big] \\ &+ \frac{\beta\eta_l^2}{2}\mathbb{E}\Big[\Big\|\sum_{i=1}^{n} w_i^e \sum_{k=0}^{E_i-1}\big(\tilde{g}_i(\boldsymbol{\theta}_{i,k}^e) - \nabla f(\boldsymbol{\theta}^e)\big)\Big\|^2\Big], \end{aligned} \tag{31}$$

where the constant inequality in the first line is achieved from our assumption that $\eta_l \leq \frac{1}{6\beta E_i}$ (and consequently: $\eta_l \leq \frac{1}{6\beta\bar{E}_l^e}$):

$$\begin{aligned}\frac{1}{2}\eta_l(1-\beta\eta_l\bar{E}_l^e) - \eta_l\bar{E}_l^e(1-\frac{\beta\eta_l}{2}\bar{E}_l^e) &= -\eta_l\left(\bar{E}_l^e - \frac{1}{2} - \frac{\beta\eta_l}{2}\bar{E}_l^{e^2} + \frac{\beta\eta_l\bar{E}_l^e}{2}\right) \\ &\leq -\eta_l\left(\frac{11\bar{E}_l^e-6}{12} + \frac{\beta\eta_l\bar{E}_l^e}{2}\right) \\ &\leq -\eta_l\frac{11\bar{E}_l^e-6}{12}. \end{aligned} \tag{32}$$

Therefore,

$$\begin{aligned}\mathbb{E}\big[f(\boldsymbol{\theta}^{e+1})\big] \leq \mathbb{E}\big[f(\boldsymbol{\theta}^{e})\big] &- \eta_l\frac{11\bar{E}_l^e-6}{12}\mathbb{E}\big[\|\nabla f(\boldsymbol{\theta}^e)\|^2\big] \\ &+ \frac{1}{2}\eta_l(1-\beta\eta_l\bar{E}_l^e)\mathbb{E}\Big[\Big\|\sum_{i=1}^{n} w_i^e \sum_{k=0}^{E_i-1}\big(\nabla f_i(\boldsymbol{\theta}_{i,k}^e) - \nabla f(\boldsymbol{\theta}^e)\big)\Big\|^2\Big] \\ &+ \frac{\beta\eta_l^2}{2}\mathbb{E}\Big[\Big\|\sum_{i=1}^{n} w_i^e \sum_{k=0}^{E_i-1}\big(\tilde{g}_i(\boldsymbol{\theta}_{i,k}^e) - \nabla f(\boldsymbol{\theta}^e)\big)\Big\|^2\Big]. \end{aligned} \tag{33}$$

Now, we use the relaxed triangle inequality $\|a+b\|^2 \leq 2(\|a\|^2+\|b\|^2)$ for the last line above:

$$\begin{aligned}\mathbb{E}\big[f(\boldsymbol{\theta}^{e+1})\big] \leq \mathbb{E}\big[f(\boldsymbol{\theta}^{e})\big] &- \eta_l\frac{11\bar{E}_l^e-6}{12}\mathbb{E}\big[\|\nabla f(\boldsymbol{\theta}^e)\|^2\big] \\ &+ \frac{1}{2}\eta_l(1-\beta\eta_l\bar{E}_l^e)\underbrace{\mathbb{E}\Big[\Big\|\sum_{i=1}^{n} w_i^e \sum_{k=0}^{E_i-1}\big(\nabla f_i(\boldsymbol{\theta}_{i,k}^e) - \nabla f(\boldsymbol{\theta}^e)\big)\Big\|^2\Big]}_{\mathcal{B}} \\ &+ \beta\eta_l^2\underbrace{\mathbb{E}\Big[\Big\|\sum_{i=1}^{n} w_i^e \sum_{k=0}^{E_i-1}\big(\tilde{g}_i(\boldsymbol{\theta}_{i,k}^e) - \nabla f_i(\boldsymbol{\theta}_{i,k}^e)\big)\Big\|^2\Big]}_{\mathcal{A}} + \beta\eta_l^2\underbrace{\mathbb{E}\Big[\Big\|\sum_{i=1}^{n} w_i^e \sum_{k=0}^{E_i-1}\big(\nabla f_i(\boldsymbol{\theta}_{i,k}^e) - \nabla f(\boldsymbol{\theta}^e)\big)\Big\|^2\Big]}_{\mathcal{B}} \end{aligned} \tag{34}$$

Now, we bound each of the terms $\mathcal{A}$ and $\mathcal{B}$ separately:

$$\begin{aligned}
\mathcal{A} &\leq \mathbb{E}\Big[\Big(\sum_{i=1}^{n} w_i^e \sum_{k=0}^{E_i-1} \big\|\tilde{g}_i(\boldsymbol{\theta}_{i,k}^e) - \nabla f_i(\boldsymbol{\theta}_{i,k}^e)\big\|\Big)^2\Big] \leq \mathbb{E}\Big[\sum_{i=1}^{n}(w_i^e)^2 \times \sum_{i=1}^{n}\Big(\sum_{k=0}^{E_i-1} \big\|\tilde{g}_i(\boldsymbol{\theta}_{i,k}^e) - \nabla f_i(\boldsymbol{\theta}_{i,k}^e)\big\|\Big)^2\Big] \\
&= \mathbb{E}\Big[\|\mathbf{w}^e\|^2 \sum_{i=1}^{n}\Big(\sum_{k=0}^{E_i-1} \big\|\tilde{g}_i(\boldsymbol{\theta}_{i,k}^e) - \nabla f_i(\boldsymbol{\theta}_{i,k}^e)\big\|\Big)^2\Big] = \mathbb{E}\Big[\sum_{i=1}^{n}\Big(\sum_{k=0}^{E_i-1} \big\|\tilde{g}_i(\boldsymbol{\theta}_{i,k}^e) - \nabla f_i(\boldsymbol{\theta}_{i,k}^e)\big\|\Big)^2\Big] \\
&\leq \sum_{i=1}^{n} E_i \sum_{k=0}^{E_i-1} \mathbb{E}\Big[\Big\|\tilde{g}_i(\boldsymbol{\theta}_{i,k}^e) - \nabla f_i(\boldsymbol{\theta}_{i,k}^e)\Big\|^2\Big] \leq \sum_{i=1}^{n} E_i^2 \sigma_{i,\tilde{g}}^2, \qquad (35)
\end{aligned}$$

where in the first and second inequalities, we used Cauchy-Schwarz inequality. In the last inequality, we used Equation (23). Similarly, we can bound $\mathcal{B}$:

$$\begin{aligned}
\mathcal{B} &= \mathbb{E}\Big[\Big\|\sum_{i=1}^{n} w_i^e \sum_{k=0}^{E_i-1} \big(\nabla f_i(\boldsymbol{\theta}_{i,k}^e) - \nabla f(\boldsymbol{\theta}^e)\big)\Big\|^2\Big] = \mathbb{E}\Big[\Big\|\sum_{i=1}^{n} w_i^e \sum_{k=0}^{E_i-1} \nabla f_i(\boldsymbol{\theta}_{i,k}^e) - \sum_{i=1}^{n} w_i^e \sum_{k=0}^{E_i-1} \nabla f(\boldsymbol{\theta}^e)\Big\|^2\Big] \\
&= \mathbb{E}\Big[\Big\|\sum_{i=1}^{n} w_i^e \sum_{k=0}^{E_i-1} \nabla f_i(\boldsymbol{\theta}_{i,k}^e) - \underbrace{\big(\sum_{i=1}^{n} w_i^e E_i\big)}_{\bar{E}_l^e} \nabla f(\boldsymbol{\theta}^e)\Big\|^2\Big] = \mathbb{E}\Big[\Big\|\Big(\sum_{i=1}^{n} w_i^e \sum_{k=0}^{E_i-1} \nabla f_i(\boldsymbol{\theta}_{i,k}^e)\Big) - \bar{E}_l^e \nabla f(\boldsymbol{\theta}^e)\Big\|^2\Big]. \qquad (36)
\end{aligned}$$

Let us also define $\Delta_i^e := w_i^e - \lambda_i$ **for client** i to be the difference between the aggregation weight of client i in round e (w_i^e) and its corresponding aggregation weights in the global objective function $f(\boldsymbol{\theta})$ (λ_i). With this definition and that $\nabla f(\boldsymbol{\theta}^e) = \sum_{i=1}^{n} \lambda_i \nabla f_i(\boldsymbol{\theta}^e)$, we have:

$$\begin{aligned}
\mathcal{B} &= \mathbb{E}\Big[\Big\|\Big(\sum_{i=1}^{n} \Delta_i^e \sum_{k=0}^{E_i-1} \nabla f_i(\boldsymbol{\theta}_{i,k}^e)\Big) + \Big(\sum_{i=1}^{n} \lambda_i \sum_{k=0}^{E_i-1} \nabla f_i(\boldsymbol{\theta}_{i,k}^e)\Big) - \Big(\sum_{i=1}^{n} \lambda_i \bar{E}_l^e \nabla f_i(\boldsymbol{\theta}^e)\Big)\Big\|^2\Big] \\
&\leq \underbrace{2\mathbb{E}\Big[\Big\|\sum_{i=1}^{n} \Delta_i^e \sum_{k=0}^{E_i-1} \nabla f_i(\boldsymbol{\theta}_{i,k}^e)\Big\|^2\Big]}_{\mathcal{C}} + \underbrace{2\mathbb{E}\Big[\Big\|\Big(\sum_{i=1}^{n} \lambda_i \sum_{k=0}^{E_i-1} \nabla f_i(\boldsymbol{\theta}_{i,k}^e)\Big) - \Big(\sum_{i=1}^{n} \lambda_i \bar{E}_l^e \nabla f_i(\boldsymbol{\theta}^e)\Big)\Big\|^2\Big]}_{\mathcal{D}}. \qquad (37)
\end{aligned}$$

Now, we bound each of the terms $\mathcal{C}$ and $\mathcal{D}$, separately:

$$\begin{aligned}
\mathcal{C} &= 2\mathbb{E}\Big[\Big\|\sum_{i=1}^{n} \Delta_i^e \sum_{k=0}^{E_i-1} \nabla f_i(\boldsymbol{\theta}_{i,k}^e)\Big\|^2\Big] \\
&\leq 4\mathbb{E}\Big[\Big\|\sum_{i=1}^{n} \Delta_i^e \sum_{k=0}^{E_i-1} \big(\nabla f_i(\boldsymbol{\theta}_{i,k}^e) - \nabla f_i(\boldsymbol{\theta}^e)\big)\Big\|^2\Big] + 4\mathbb{E}\Big[\Big\|\sum_{i=1}^{n} E_i \Delta_i^e \nabla f_i(\boldsymbol{\theta}^e)\Big\|^2\Big] \\
&\leq 4\mathbb{E}\Big[\big(\sum_{i=1}^{n} E_i\big) \sum_{i=1}^{n} \sum_{k=0}^{E_i-1} |\Delta_i^e|^2 \|\nabla f_i(\boldsymbol{\theta}_{i,k}^e) - \nabla f_i(\boldsymbol{\theta}^e)\|^2\Big] + 4\mathbb{E}\Big[n \sum_{i=1}^{n} \Big\|E_i \Delta_i^e \nabla f_i(\boldsymbol{\theta}^e)\Big\|^2\Big] \\
&\leq 4\big(\sum_{i=1}^{n} E_i\big)\beta^2 \sum_{i=1}^{n} \sum_{k=0}^{E_i-1} |\Delta_i^e|^2 \mathbb{E}[\|\boldsymbol{\theta}_{i,k}^e - \boldsymbol{\theta}^e\|^2] + 4nL_0^2 \sum_{i=1}^{n} E_i^2 \mathbb{E}[|\Delta_i^e|^2] \\
&\leq 4\beta^2\big(\sum_{i=1}^{n} E_i\big) \sum_{i=1}^{n} \sum_{k=0}^{E_i-1} \mathbb{E}[\|\boldsymbol{\theta}_{i,k}^e - \boldsymbol{\theta}^e\|^2] + 4nL_0^2 \sum_{i=1}^{n} E_i^2 \mathbb{E}[|\Delta_i^e|^2], \qquad (38)
\end{aligned}$$

where in the third line, we have used relaxed triangle inequality, and in the fourth line, we have used β-smoothness and L_0-Lipschitz continuity of f_i. Also, in the last line we used $|\Delta_i^e| \leq 1$. Similarly:

$$\begin{aligned}
\mathcal{D} &= 2\mathbb{E}\Big[\Big\| \sum_{i=1}^{n} \lambda_i \Big(\sum_{k=0}^{E_i-1} \nabla f_i(\boldsymbol{\theta}_{i,k}^e) - \bar{E}_l^e \nabla f_i(\boldsymbol{\theta}^e) \Big) \Big\|^2 \Big] \\
&\leq 2\|\boldsymbol{\lambda}\|^2 \sum_{i=1}^{n} \mathbb{E}\Big[\Big\| \sum_{k=0}^{E_i-1} \nabla f_i(\boldsymbol{\theta}_{i,k}^e) - \bar{E}_l^e \nabla f_i(\boldsymbol{\theta}^e) \Big\|^2 \Big] \\
&\leq 2\|\boldsymbol{\lambda}\|^2 \sum_{i=1}^{n} \mathbb{E}\Big[\Big\| \sum_{k=0}^{E_i-1} \big(\nabla f_i(\boldsymbol{\theta}_{i,k}^e) - \nabla f_i(\boldsymbol{\theta}^e)\big) + \big(E_i - \bar{E}_l^e\big) \nabla f_i(\boldsymbol{\theta}^e) \Big\|^2 \Big] \\
&\leq 4\|\boldsymbol{\lambda}\|^2 \sum_{i=1}^{n} \mathbb{E}\Big[\Big\| \sum_{k=0}^{E_i-1} \nabla f_i(\boldsymbol{\theta}_{i,k}^e) - \nabla f_i(\boldsymbol{\theta}^e) \Big\|^2 + \big(E_i - \bar{E}_l^e\big)^2 \underbrace{\big\|\nabla f_i(\boldsymbol{\theta}^e)\big\|^2}_{\leq L_0^2} \Big] \\
&\leq 4\beta^2 \|\boldsymbol{\lambda}\|^2 \sum_{i=1}^{n} E_i \sum_{k=0}^{E_i-1} \mathbb{E}\big[\big\|\boldsymbol{\theta}_{i,k}^e - \boldsymbol{\theta}^e\big\|^2\big] + 4L_0^2 \|\boldsymbol{\lambda}\|^2 \sum_{i=1}^{n} \mathbb{E}[(E_i - \bar{E}_l^e)^2]. \qquad (39)
\end{aligned}$$

In the first inequality, we used convexity of the norm function, and Cauchy-Schwarz inequality. Hence, by plugging the bounds above on $\mathcal{C}$ and $\mathcal{D}$ into Equation (37), we get:

$$\mathcal{B} \leq 4\beta^2 (1 + \sum_{i=1}^{n} E_i) \Big(\sum_{i=1}^{n} E_i \sum_{k=0}^{E_i-1} \mathbb{E}\big[\big\|\boldsymbol{\theta}_{i,k}^e - \boldsymbol{\theta}^e\big\|^2\big] \Big) + 4L_0^2 \Big(n \sum_{i=1}^{n} E_i^2 \mathbb{E}[|\Delta_i^e|^2] + \|\boldsymbol{\lambda}\|^2 \sum_{i=1}^{n} \mathbb{E}[(E_i - \bar{E}_l^e)^2] \Big) \qquad (40)$$

In the following, we first simplify Equation (34), and then, we plugg the bounds above on $\mathcal{A}$ and $\mathcal{B}$ in it. We have:

$$\begin{aligned}
\mathbb{E}\big[f(\boldsymbol{\theta}^{e+1})\big] \leq\ & \mathbb{E}\big[f(\boldsymbol{\theta}^e)\big] - \eta_l \frac{11\bar{E}_l^e - 6}{12} \mathbb{E}\big[\|\nabla f(\boldsymbol{\theta}^e)\|^2\big] \\
&+ \beta\eta_l^2 \underbrace{\mathbb{E}\Big[\Big\| \sum_{i=1}^{n} w_i^e \sum_{k=0}^{E_i-1} \big(\tilde{g}_i(\boldsymbol{\theta}_{i,k}^e) - \nabla f_i(\boldsymbol{\theta}_{i,k}^e)\big) \Big\|^2 \Big]}_{\mathcal{A}} \\
&+ \underbrace{(\beta\eta_l^2 + \frac{1}{2}\eta_l(1 - \beta\eta_l \bar{E}_l^e))}_{< \frac{2}{3}\eta_l} \underbrace{\mathbb{E}\Big[\Big\| \sum_{i=1}^{n} w_i^e \sum_{k=0}^{E_i-1} \big(\nabla f_i(\boldsymbol{\theta}_{i,k}^e) - \nabla f(\boldsymbol{\theta}^e)\big) \Big\|^2 \Big]}_{\mathcal{B}}, \qquad (41)
\end{aligned}$$

where from the assumption $\eta_l \leq \frac{1}{6\beta E_i}$, we get to $\frac{\beta\eta_l^2}{2} \leq \frac{\eta_l}{12}$. Hence:

$$\beta\eta_l^2 + \frac{1}{2}\eta_l(1 - \beta\eta_l \bar{E}_l^e) = \beta\eta_l^2(1 - \frac{\bar{E}_l^e}{2}) + \frac{\eta_l}{2} \leq \frac{\beta\eta_l^2}{2} + \frac{\eta_l}{2} \leq \frac{\eta_l}{12} + \frac{\eta_l}{2} < \frac{2\eta_l}{3}. \qquad (42)$$

Therefore, by plugging in the bounds on $\mathcal{A}$ and $\mathcal{B}$, we have:

$$\begin{aligned}
\mathbb{E}\big[f(\boldsymbol{\theta}^{e+1})\big] \leq\ & \mathbb{E}\big[f(\boldsymbol{\theta}^e)\big] - \eta_l \frac{11\bar{E}_l^e - 6}{12} \mathbb{E}\big[\|\nabla f(\boldsymbol{\theta}^e)\|^2\big] + \beta\eta_l^2 \sum_{i=1}^{n} E_i^2 \sigma_{i,\tilde{g}}^2 \\
&+ \Big(\frac{8}{3}\beta^2 \eta_l (1 + \sum_{i=1}^{n} E_i) \Big(\sum_{i=1}^{n} E_i \sum_{k=0}^{E_i-1} \mathbb{E}\big[\big\|\boldsymbol{\theta}_{i,k}^e - \boldsymbol{\theta}^e\big\|^2\big] \Big) \Big) \\
&+ \Big(\frac{8}{3} L_0^2 \eta_l \Big(n \sum_{i=1}^{n} E_i^2 \mathbb{E}[|\Delta_i^e|^2] + \|\boldsymbol{\lambda}\|^2 \sum_{i=1}^{n} \mathbb{E}[(E_i - \bar{E}_l^e)^2] \Big) \Big). \qquad (43)
\end{aligned}$$

We now have the following lemma to bound local drift of clients during each communication round e:

Lemma E.1 (Bounded local drifts). *Suppose Assumption D.1 holds. The local drift happening at client i during communication round e is bounded:*

$$\xi_i^e := \sum_{k=0}^{E_i-1} \mathbb{E}\big[\|\boldsymbol{\theta}_{i,k}^e - \boldsymbol{\theta}^e\|^2\big] \leq (\texttt{cte}-2)E_i^2\eta_l^2\left(\sigma_{i,\tilde{g}^2} + 6E_i\sigma^2 + 6E_i\mathbb{E}[\|\nabla f(\boldsymbol{\theta}^e)\|^2]\right), \tag{44}$$

where $\texttt{cte}$ *is the mathematical constant* e.

Proof. From $\boldsymbol{\theta}_{i,0}^e = \boldsymbol{\theta}^e$, we only need to focus on $E_i \geq 2$. We have:

$$\begin{aligned}\mathbb{E}\|\boldsymbol{\theta}_{i,k}^e - \boldsymbol{\theta}^e\|^2 &= \mathbb{E}[\|\boldsymbol{\theta}_{i,k-1}^e - \eta_l\tilde{g}_i(\boldsymbol{\theta}_{i,k-1}^e) - \boldsymbol{\theta}^e\|^2] \\ &\leq \mathbb{E}[\|\boldsymbol{\theta}_{i,k-1}^e - \eta_l\nabla f_i(\boldsymbol{\theta}_{i,k-1}^e) - \boldsymbol{\theta}^e\|^2] + \eta_l^2\sigma_{i,\tilde{g}}^2\end{aligned} \tag{45}$$

where the inequality comes from Lemma D.4. The first term on the right side of the above inequality can be bounded as:

$$\mathbb{E}[\|\boldsymbol{\theta}_{i,k-1}^e - \eta_l\nabla f_i(\boldsymbol{\theta}_{i,k-1}^e) - \boldsymbol{\theta}^e\|^2] \leq \left(1 + \frac{1}{2E_i-1}\right)\mathbb{E}[\|\boldsymbol{\theta}_{i,k-1}^e - \boldsymbol{\theta}^e\|^2] + 2E_i\eta_l^2\mathbb{E}\|[\nabla f_i(\boldsymbol{\theta}_{i,k-1}^e)\|^2], \tag{46}$$

where we have used Lemma D.3. Now, we bound the last term in the above inequality. We have:

$$\nabla f_i(\boldsymbol{\theta}_{i,k-1}^e) = (\nabla f_i(\boldsymbol{\theta}_{i,k-1}^e) - \nabla f_i(\boldsymbol{\theta}^e)) + (\nabla f_i(\boldsymbol{\theta}^e) - \nabla f(\boldsymbol{\theta}^e)) + \nabla f(\boldsymbol{\theta}^e), \tag{47}$$

By using relaxed triangle inequality (Lemma D.3) and Assumption D.1, we get:

$$\begin{aligned}\|\nabla f_i(\boldsymbol{\theta}_{i,k-1}^e)\|^2 &= 3\|\nabla f_i(\boldsymbol{\theta}_{i,k-1}^e) - \nabla f_i(\boldsymbol{\theta}^e)\|^2 + 3\|\nabla f_i(\boldsymbol{\theta}^e) - \nabla f(\boldsymbol{\theta}^e)\|^2 + 3\|\nabla f(\boldsymbol{\theta}^e)\|^2 \\ &\leq 3\beta^2\|\boldsymbol{\theta}_{i,k-1}^e - \boldsymbol{\theta}^e\|^2 + 3\sigma^2 + 3\|\nabla f(\boldsymbol{\theta})\|^2.\end{aligned} \tag{48}$$

Now, we can rewrite Equation (46) and then Equation (45):

$$\begin{aligned}\mathbb{E}\|\boldsymbol{\theta}_{i,k}^e - \boldsymbol{\theta}^e\|^2 \leq &\underbrace{\left(1 + \frac{1}{2E_i-1} + 6E_i\beta^2\eta_l^2\right)}_{\leq 1+\frac{1}{E_i}}\mathbb{E}[\|\boldsymbol{\theta}_{i,k-1}^e - \boldsymbol{\theta}^e\|^2] \\ &+ \eta_l^2(6E_i\sigma^2 + \sigma_{i,\tilde{g}}^2) + 6E_i\eta_l^2\mathbb{E}\|\nabla f(\boldsymbol{\theta}^e)\|^2 \\ &\leq (1 + \frac{1}{E_i})\mathbb{E}[\|\boldsymbol{\theta}_{i,k-1}^e - \boldsymbol{\theta}^e\|^2] + \eta_l^2(6E_i\sigma^2 + \sigma_{i,\tilde{g}}^2) + 6E_i\eta_l^2\mathbb{E}[\|\nabla f(\boldsymbol{\theta}^e)\|^2]\end{aligned} \tag{49}$$

From the inequality above and that $\mathbb{E}\|\boldsymbol{\theta}_{i,0}^e - \boldsymbol{\theta}^e\|^2 = 0$, we have:

$$\begin{aligned}&\mathbb{E}\|\boldsymbol{\theta}_{i,1}^e - \boldsymbol{\theta}^e\|^2 \leq \gamma \\ &\mathbb{E}\|\boldsymbol{\theta}_{i,2}^e - \boldsymbol{\theta}^e\|^2 \leq (1 + \frac{1}{E_i})\gamma + \gamma \\ &\mathbb{E}\|\boldsymbol{\theta}_{i,3}^e - \boldsymbol{\theta}^e\|^2 \leq (1 + \frac{1}{E_i})^2\gamma + (1 + \frac{1}{E_i})\gamma + \gamma \\ &\ldots \\ &\mathbb{E}\|\boldsymbol{\theta}_{i,k}^e - \boldsymbol{\theta}^e\|^2 \leq (1 + \frac{1}{E_i})^{(k-1)}\gamma + \ldots + (1 + \frac{1}{E_i})^2\gamma + (1 + \frac{1}{E_i})\gamma + \gamma,\end{aligned} \tag{50}$$

where $\gamma = \eta_l^2(6E_i\sigma^2 + \sigma_{i,\tilde{g}}^2) + 6E_i\eta_l^2\mathbb{E}[\|\nabla f(\boldsymbol{\theta}^e)\|^2]$. By using $1 + q + \cdots + q^{n-1} = \frac{q^n-1}{q-1}$, we get:

$$\mathbb{E}\|\boldsymbol{\theta}_{i,k}^e - \boldsymbol{\theta}^e\|^2 \leq E_i\bigg(\big(1 + \frac{1}{E_i}\big)^k - 1\bigg)\big(\eta_l^2(6E_i\sigma^2 + \sigma_{i,\tilde{g}}^2) + 6E_i\eta_l^2\mathbb{E}[\|\nabla f(\boldsymbol{\theta}^e)\|^2]\big). \tag{51}$$

Therefore, we have:

$$\begin{aligned}\sum_{k=0}^{E_i-1}\mathbb{E}\|\boldsymbol{\theta}^e_{i,k}-\boldsymbol{\theta}^e\|^2 &\leq E_i^2\bigg(\underbrace{\big(1+\frac{1}{E_i}\big)^{E_i}}_{\leq\texttt{cte}}-2\bigg)\big(\eta_l^2(6E_i\sigma^2+\sigma^2_{i,\tilde{g}})+6E_i\eta_l^2\mathbb{E}[\|\nabla f(\boldsymbol{\theta}^e)\|^2]\big)\\ &\leq(\texttt{cte}-2)E_i^2\eta_l^2\big(6E_i\sigma^2+\sigma^2_{i,\tilde{g}}+6E_i\mathbb{E}[\|\nabla f(\boldsymbol{\theta}^e)\|^2]\big),\end{aligned}\tag{52}$$

where $E_i \geq 2$ and $\texttt{cte}$ above is the mathematical constant e. □

We can now plug the bound on local drifts into Equation (43) and get:

$$\begin{aligned}\mathbb{E}\big[f(\boldsymbol{\theta}^{e+1})\big] \leq \mathbb{E}\big[f(\boldsymbol{\theta}^e)\big]-\eta_l\underbrace{\bigg(\frac{11\bar{E}^e_l-6}{12}-12\beta^2\eta_l^2(1+\sum_{i=1}^n E_i)\big(\sum_{i=1}^n E_i^4\big)\bigg)}_{\geq\frac{11\bar{E}^e_l-7}{12}}\mathbb{E}\big[\|\nabla f(\boldsymbol{\theta}^e)\|^2\big]\\ +6\beta^2\eta_l^3(1+\sum_{i=1}^n E_i)\bigg(2\sum_{i=1}^n E_i^4\sigma^2+\frac{1}{3}\sum_{i=1}^n E_i^3\sigma^2_{i,\tilde{g}}\bigg)+\beta\eta_l^2\sum_{i=1}^n E_i^2\sigma^2_{i,\tilde{g}}\\ +\frac{8}{3}L_0^2\eta_l\bigg(n\sum_{i=1}^n E_i^2\mathbb{E}[(w_i^e-\lambda_i)^2]+\|\boldsymbol{\lambda}\|^2\sum_{i=1}^n\mathbb{E}[(E_i-\bar{E}^e_l)^2]\bigg),\end{aligned}\tag{53}$$

where we have used the second condition on η_l in the first line to bound the multiplicative factor.

Hence, we have:

$$\begin{aligned}&\mathbb{E}\big[f(\boldsymbol{\theta}^{e+1})\big]\leq\\ &\mathbb{E}\big[f(\boldsymbol{\theta}^e)\big]-\eta_l\bigg(\frac{11\bar{E}^e_l-7}{12}\bigg)\mathbb{E}\big[\|\nabla f(\boldsymbol{\theta}^e)\|^2\big]\\ &+\eta_l\underbrace{\bigg(6\beta^2\eta_l^2(1+\sum_{i=1}^n E_i)\bigg(2\sum_{i=1}^n E_i^4\sigma^2+\frac{1}{3}\sum_{i=1}^n E_i^3\sigma^2_{i,\tilde{g}}\bigg)+\beta\eta_l\sum_{i=1}^n E_i^2\sigma^2_{i,\tilde{g}}\bigg)}_{\Psi_\sigma}\\ &+\eta_l\underbrace{\frac{8L_0^2}{3}\bigg(n\sum_{i=1}^n E_i^2\mathbb{E}[(w_i^e-\lambda_i)^2]+\|\boldsymbol{\lambda}\|^2\sum_{i=1}^n\mathbb{E}[(E_i-\bar{E}^e_l)^2]\bigg)}_{\Psi_p}.\end{aligned}\tag{54}$$

Remind that $\bar{E}^e_l=\sum_{i=1}^n w_i^e E_i$ is a weighted average of clients' number of local gradient steps. From above, we have:

$$\eta_l(\frac{11\bar{E}^e_l-7}{12})\mathbb{E}\|\nabla f(\boldsymbol{\theta}^e)\|^2\leq\mathbb{E}[f(\boldsymbol{\theta}^e)-f(\boldsymbol{\theta}^{e+1})]+(\Psi_\sigma+\Psi_p)\eta_l.\tag{55}$$

We can now replace $\bar{E}^e_l$, which is a weighted average of $\{E_i\}_{i=1}^n$ in round e, with $E_l^{\texttt{min}}=\min_i\{E_i\}_{i=1}^n$, and the inequality still holds:

$$\eta_l(\frac{11E_l^{\texttt{min}}-7}{12})\mathbb{E}\|\nabla f(\boldsymbol{\theta}^e)\|^2\leq\mathbb{E}[f(\boldsymbol{\theta}^e)-f(\boldsymbol{\theta}^{e+1})]+(\Psi_p+\Psi_\sigma)\eta_l.\tag{56}$$

By summing both sides of the above inequality over $e=0,\ldots,E-1$ and dividing by E, we get:

$$\min_{0\leq e\leq E-1}\mathbb{E}[\|\nabla f(\boldsymbol{\theta}^e)\|^2]\leq\frac{12}{11E_l^{\texttt{min}}-7}\bigg(\frac{f(\boldsymbol{\theta}^0)-f^*}{E\eta_l}+\Psi_\sigma+\Psi_p\bigg),\tag{57}$$

which completes the proof. □

F. Detailed results

F.1. Test accuracy comparison

In Table 12 to 15, we report the detailed test accuracy values for all algorithms, on all datasets and privacy distributions we study in this work. The results show that Robust-HDP is consistently outperforming the state-of-the-art algorithms across various datasets.

Table 12. Comparison of different algorithms (on MNIST, $E = 200$). FedAvg achieves 98.6%.

distr alg	Dist1	Dist2	Dist3	Dist4	Dist5	Dist6	Dist7	Dist8	Dist9
WeiAvg (Liu et al., 2021a)	**90.08**	88.29	89.74	88.20	84.94	81.40	84.43	78.71	81.38
PFA (Liu et al., 2021a)	88.24	87.93	88.35	88.32	85.65	82.16	83.71	80.25	78.51
DPFedAvg (Noble et al., 2021)	84.24	82.84	83.50	80.43	83.02	75.69	**85.71**	70.58	80.49
minimum ϵ (Liu et al., 2021a)	77.80	74.86	74.86	71.75	68.42	34.32	77.62	56.10	68.44
Robust-HDP	89.83	**90.71**	**89.83**	**89.38**	**87.52**	**84.60**	84.03	**81.19**	**81.52**

Table 13. Comparison of different algorithms (on FMNIST, $E = 200$). FedAvg achieves 90.28%.

distr alg	Dist1	Dist2	Dist3	Dist4	Dist5	Dist6	Dist7	Dist8	Dist9
WeiAvg (Liu et al., 2021a)	**77.65**	**78.30**	**75.92**	**77.10**	72.38	64.15	66.80	66.86	64.79
PFA (Liu et al., 2021a)	75.03	74.85	72.90	**77.10**	72.44	71.08	66.30	67.89	64.69
DPFedAvg (Noble et al., 2021)	74.12	71.68	71.97	68.10	70.20	62.46	64.15	65.87	65.50
minimum ϵ (Liu et al., 2021a)	73.15	64.26	64.26	62.60	64.35	28.66	65.13	58.44	66.36
Robust-HDP	75.13	76.25	75.04	76.19	**73.80**	**71.30**	**66.85**	**68.32**	**66.96**

Table 14. Comparison of different algorithms (on CIFAR10, $E = 200$). FedAvg achieves 73.55%.

distr alg	Dist1	Dist2	Dist3	Dist4	Dist5	Dist6	Dist7	Dist8	Dist9
WeiAvg (Liu et al., 2021a)	31.18	31.18	29.65	27.74	24.25	19.91	21.93	18.91	20.64
PFA (Liu et al., 2021a)	26.91	**32.68**	25.19	29.21	21.63	18.93	20.63	16.27	15.75
DPFedAvg (Noble et al., 2021)	31.51	21.51	22.28	20.50	21.25	15.19	18.27	16.45	18.63
minimum ϵ (Liu et al., 2021a)	26.20	16.71	16.45	15.86	14.23	10.51	13.35	13.32	14.11
Robust-HDP	**31.97**	31.70	**32.0**	**30.60**	**24.86**	**23.61**	**24.10**	**19.02**	**22.05**

F.2. ablation study on privacy level and number of clients

The results in Table 16 and Table 17 show the detailed results for the ablation study on privacy level and number of clients, reported in Figure 5 (left and middle figures, respectively). The values are the mean and standard deviation of average test accuracy across clients over three different runs.

Table 15. Comparison of different algorithms (on CIFAR100, $E = 200$). FedAvg achieves 61.80%.

distr alg	Dist1	Dist2	Dist3	Dist4	Dist5	Dist6	Dist7	Dist8	Dist9
WeiAvg (Liu et al., 2021a)	**35.91**	**36.23**	32.61	30.92	29.42	27.37	27.26	27.03	26.57
PFA (Liu et al., 2021a)	34.21	35.86	30.12	29.45	26.95	31.27	24.35	21.29	18.05
DPFedAvg (Noble et al., 2021)	31.34	31.30	25.01	26.74	24.96	21.27	21.72	17.36	17.71
minimum ϵ (Liu et al., 2021a)	27.61	27.25	24.91	25.02	25.07	21.21	21.46	17.42	17.50
Robust-HDP	33.35	33.38	**33.46**	**35.03**	**31.58**	**31.33**	**30.27**	**30.71**	**29.21**

Table 16. Detailed results for ablation study on the privacy level of clients in Figure 5, left.

distr alg	Dist2	Dist4	Dist6	Dist8
WeiAvg (Liu et al., 2021a)	$88.29_{\pm 0.67}$	$88.20_{\pm 0.52}$	$81.60_{\pm 0.58}$	$78.71_{\pm 0.67}$
PFA (Liu et al., 2021a)	$88.32_{\pm 0.85}$	$86.91_{\pm 0.97}$	$82.16_{\pm 0.99}$	$79.92_{\pm 0.86}$
DPFedAvg (Noble et al., 2021)	$82.84_{\pm 0.65}$	$80.43_{\pm 1.72}$	$74.02_{\pm 1.54}$	$70.58_{\pm 1.43}$
Robust-HDP	$\mathbf{90.71}_{\pm 0.65}$	$\mathbf{89.38}_{\pm 0.76}$	$\mathbf{85.13}_{\pm 0.68}$	$\mathbf{81.19}_{\pm 0.97}$

Table 17. Detailed results for ablation study on number of clients in Figure 5, middle.

distr alg	$n = 20$	$n = 40$	$n = 60$
WeiAvg (Liu et al., 2021a)	$81.60_{\pm 0.58}$	$75.20_{\pm 0.67}$	$63.12_{\pm 0.78}$
PFA (Liu et al., 2021a)	$82.16_{\pm 0.99}$	$73.15_{\pm 1.02}$	$66.0_{\pm 0.98}$
DPFedAvg (Noble et al., 2021)	$74.02_{\pm 1.54}$	$62.98_{\pm 1.85}$	$58.49_{\pm 1.67}$
Robust-HDP	$\mathbf{85.13}_{\pm 0.68}$	$\mathbf{76.85}_{\pm 0.75}$	$\mathbf{72.77}_{\pm 0.78}$

F.3. Precision of Robust-HDP

In this section, we investigate the precision of Robust-HDP in estimating $\{\sigma_i^2\}_{i=1}^n$ and $\{w_i^*\}_{i=1}^n$. We also check the performance of RPCA algorithm used by Robust-HDP. Figure 9 shows the eigen values of the matrices $\mathbf{M}$ and $\mathbf{L}$ on MNIST at the end of the first global communication round for when clients' privacy parameters are sampled from Dist3 (inducing less DP noise) and Dist9 (inducing more DP noise) from Table 7. We can clearly observe that most of the eigen values of $\mathbf{L}$ returned by RPCA are close to 0, especially for Dist3, i.e., RPCA has returned a low-rank matrix as the underlying low-rank matrix in $\mathbf{M}$ for both Dist3 and Dist9.

In Figure 10, we have shown the noise variance estimates $\{\hat{\sigma}_i^2\}_{i=1}^n$ and the aggregation weights $\{w_i\}_{i=1}^n$ returned by Robust-HDP, and compared them with their true (optimum) values. We have also shown the weights assigned by other baseline algorithms. Having both privacy and batch size heterogeneity, Robust-HDP assigns larger weights to clients with smaller ϵ and larger batch size (e.g., client 10, which has the largest batch size, has the largest assigned aggregation weight from Robust-HDP). The weight assignment of Robust-HDP is based on the noise estimates $\{\hat{\sigma}_i^2\}_{i=1}^n$: the larger the $\hat{\sigma}_i^2$, the smaller the assigned weight w_i. Also, as observed, the weight assignment of Robust-HDP is very close to the optimum wights $\{w_i^*\}_{i=1}^n$. In contrast, WeiAvg and PFA assign weights just based on the privacy parameters ϵ_i of clients, which is suboptimal. Similarly, DPFedAvg assings weights just based on the train set size of clients, which we assumed are uniform for the experiments in the main body of the paper and Figure 10. We have done similar comparisons in the next section (Appendix G) for other heterogeneity scenarios.

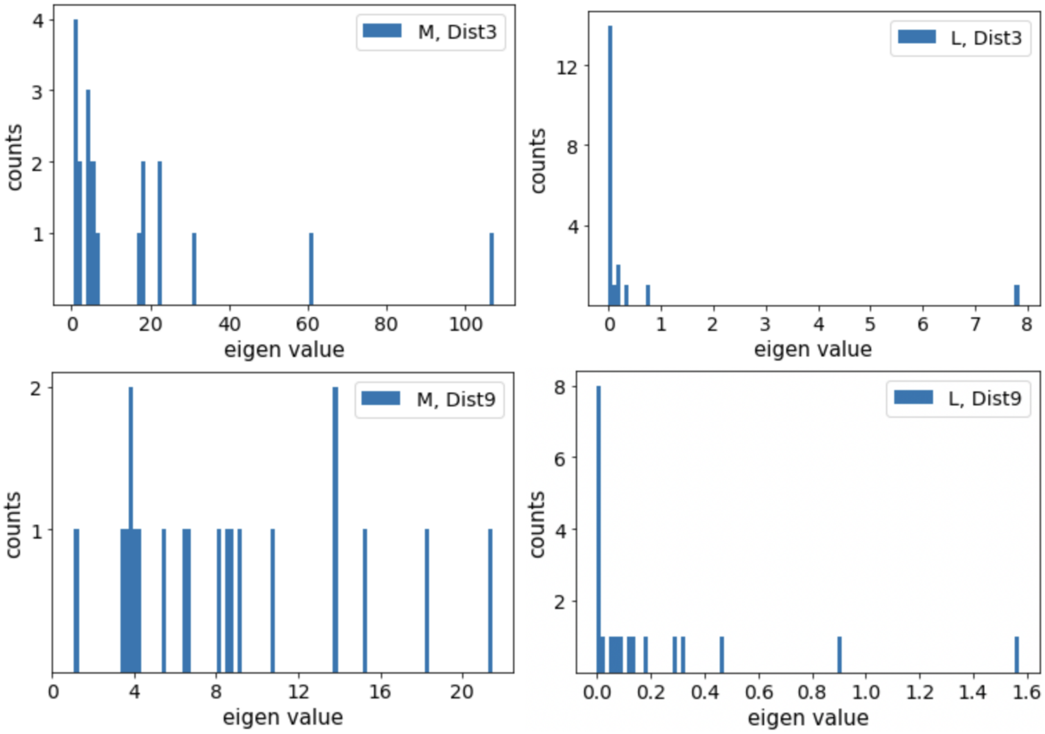

Figure 9. Comparison of the eigen values of matrices $\mathbf{M}$ (left) and $\mathbf{L}$ (right) on MNIST dataset. The concentration of eigen values in the right figures around 0 shows that the matrix $\mathbf{L}$ returned by Robust PCA, is indeed low rank, while $\mathbf{M}$ is not, due to the noise existing in clients model updates. The results for these experiments were reported in Table 12 (Robust-HDP, Dist3 and Dist9).

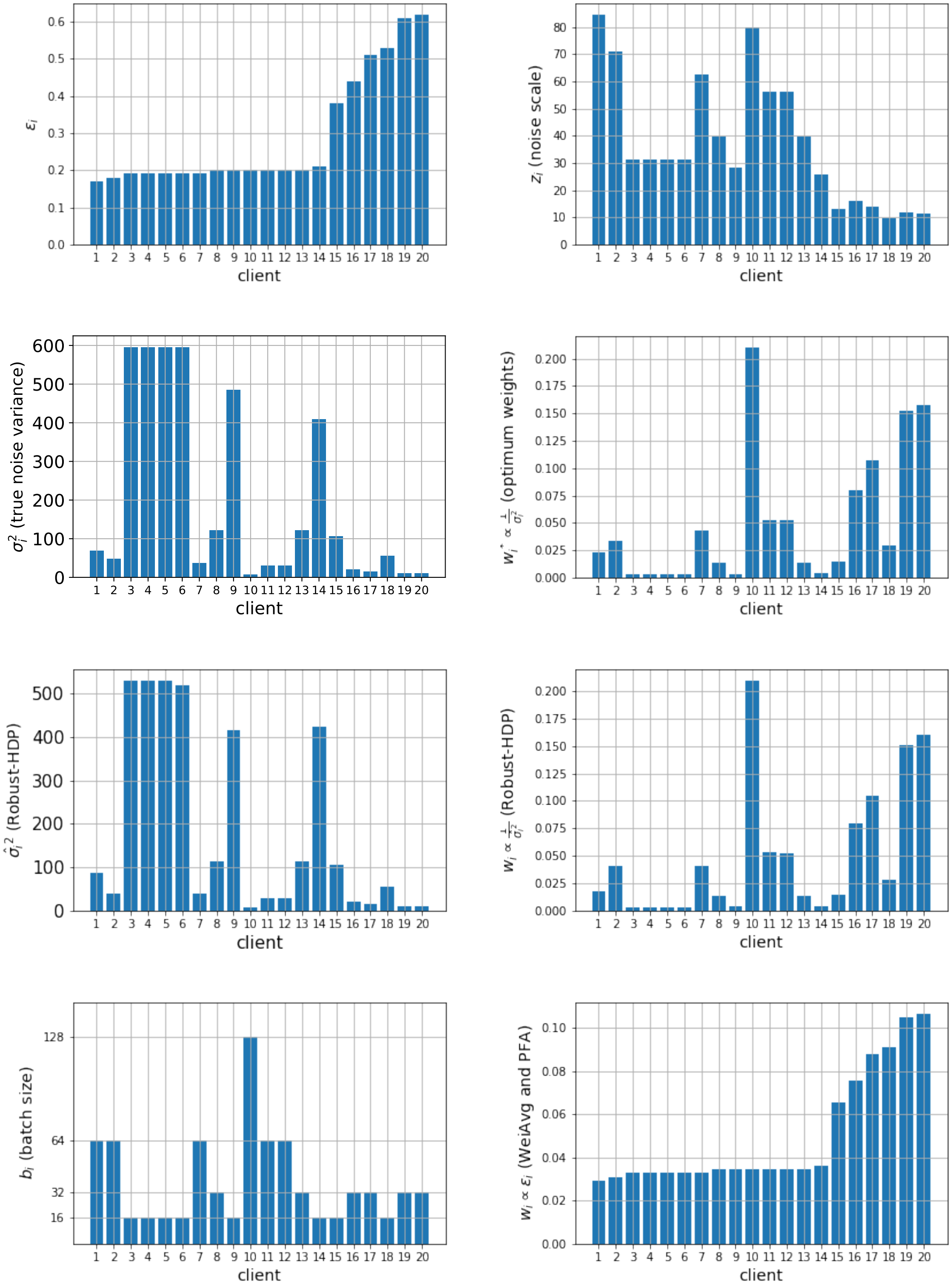

Figure 10. Comparison of weight assignments for Dist8 and MNIST with the data split in Table 6. The assigned weights by baseline algorithms show that their weight assignment strategies are not based on the noise variance in clients model updates, hence suboptimal. The results for this experiment were reported in Table 12 (Robust-HDP, Dist8).

G. Additional Experiments

So far, we assumed heterogeneous batch sizes $\{b_i\}_{i=1}^n$, heterogeneous privacy parameters $\{\epsilon_i\}_{i=1}^n$ and uniform dataset sizes $\{N_i\}_{i=1}^n$. Now, we report and discuss some extra experimental results in this section. We consider three cases:

- uniform batch sizes $\{b_i = b\}$, heterogeneous privacy parameters $\{\epsilon_i\}$ and dataset sizes $\{N_i\}$
- uniform privacy parameters $\{\epsilon_i = \epsilon\}$, heterogeneous batch sizes $\{b_i\}$ and dataset sizes $\{N_i\}$
- uniform batch sizes $\{b_i = b\}$ and uniform privacy parameters $\{\epsilon_i = \epsilon\}$, heterogeneous dataset sizes $\{N_i\}$ (corresponding to regular homogeneous `DPFL` setting, which is well-studied in the literature as a separate topic)

We run experiments on CIFAR10, as it uses a large size model and is more challenging. Unless otherwise stated, we use Dirichlet allocation (Wang et al., 2019a) to get label distribution heterogeneity for the experiments in this section. For all samples in each class k, denoted as the set $\mathcal{S}_k$, we split $\mathcal{S}_k = \mathcal{S}_{k,1} \cup \mathcal{S}_{k,2} \cup \cdots \cup \mathcal{S}_{k,n}$ into n clients ($n = 20$) according to a symmetric Dirichlet distribution $\text{Dir}(1)$. Then we gather the samples for client j as $\mathcal{S}_{1,j} \cup \mathcal{S}_{2,j} \cup \cdots \cup \mathcal{S}_{C,j}$, if we have C classes in total. This results in different dataset sizes (N_i) for different clients. After splitting the data across the clients, we fix it and run the following experiments.

G.1. uniform batch sizes $\{b_i = b\}$, heterogeneous privacy parameters $\{\epsilon_i\}$ and heterogeneous dataset sizes $\{N_i\}$

Despite the haterogeneity that may exist in the memory budgets and physical batch sizes of clients, they may use gradient accumulation (see Appendix G.5) to implement DPSGD with the same logical batch size. However, such a synchronization can happen only when the untrusted server asks clients to all use a specific logical batch size. Otherwise, if every client decides about its batch size locally, the same batch size heterogeneity that we considered in the main body of the paper will happen again. In the case of such a batch size synchronization by the server, there will be some discrepancy between the upload times of clients' model updates (as some need to use gradient accumulation with smaller physical batch sizes), which should be tolerated by the server. Having these points in mind, in this subsection, we assume such a batch size synchronization exists and we fix the logical batch size of all clients to the same value of $b = 32$ by using gradient accumulation. We also sample their privacy preference parameters $\{\epsilon_i\}$ from Table 7. In this case, our analysis in Section 3 can be rewritten as follows (as before, we use the same $\delta_i = \delta$ and $K_i = K$ for all clients):

1. Effective clipping threshold: when the clipping is indeed effective for all samples, the variance of the noisy stochastic gradient in Equation (1) can be computed as:

$$\mathbb{E}[\tilde{g}_i(\boldsymbol{\theta})] = \frac{1}{b}\sum_{j\in\mathcal{B}_i^t}\mathbb{E}[\bar{g}_{ij}(\boldsymbol{\theta})] = \frac{1}{b}\sum_{j\in\mathcal{B}_i^t} G_i(\boldsymbol{\theta}) = G_i(\boldsymbol{\theta}), \tag{58}$$

$$\sigma^2_{i,\tilde{g}} := \texttt{Var}(\tilde{g}_i(\boldsymbol{\theta})) = \frac{c^2 - \left\|G_i(\boldsymbol{\theta})\right\|^2}{b} + \frac{pc^2 z^2(\epsilon_i, \delta, \frac{b}{N_i}, K, E)}{b^2} \approx \frac{pc^2 z^2(\epsilon_i, \delta, \frac{b}{N_i}, K, E)}{b^2}, \tag{59}$$

2. Ineffective clipping threshold: when the clipping is ineffective for all samples, we have:

$$\mathbb{E}[\tilde{g}_i(\boldsymbol{\theta})] = \mathbb{E}[g_i(\boldsymbol{\theta})] = \nabla f_i(\boldsymbol{\theta}), \tag{60}$$

$$\sigma^2_{i,\tilde{g}} = \texttt{Var}(\tilde{g}_i(\boldsymbol{\theta})) = \texttt{Var}(g_i(\boldsymbol{\theta})) + \frac{p\sigma^2_{i,\texttt{DP}}}{b^2} \leq \sigma^2_{i,g} + \frac{pc^2 z^2(\epsilon_i, \delta, \frac{b}{N_i}, K, E)}{b^2},. \tag{61}$$

Finally:

$$\sigma_i^2 := \texttt{Var}(\Delta\tilde{\boldsymbol{\theta}}_i^e | \boldsymbol{\theta}^e) = K \cdot \lceil \frac{N_i}{b} \rceil \cdot \eta_l^2 \cdot \sigma^2_{i,\tilde{g}}. \tag{62}$$

Table 18. Comparison of different algorithms on CIFAR10 with uniform batch sizes $\{b_i = b\}$, heterogeneous privacy parameters $\{\epsilon_i\}$ and heterogeneous dataset sizes $\{N_i\}$. FedAvg ($E = 200$) achieves 77.58%. We have dropped the ”minimum ϵ” algorithm due to its very low performance.

distr alg	Dist1	Dist2	Dist3	Dist4	Dist5	Dist6	Dist7	Dist8	Dist9
WeiAvg (Liu et al., 2021a)	31.99	31.18	29.59	25.97	24.66	16.61	23.13	17.01	14.34
PFA (Liu et al., 2021a)	32.12	32.2	30.11	28.48	25.09	16.85	**23.34**	17.11	15.12
DPFedAvg (Noble et al., 2021)	33.89	24.16	24.62	17.50	22.71	16.76	19.52	13.81	**16.27**
Robust-HDP	**34.94**	**33.78**	**31.34**	**32.50**	**26.05**	**17.98**	23.13	**17.97**	15.79

We observe that, the amount of noise in model updates (σ_i^2) varies across clients depending on their privacy parameter ϵ_i **and dataset size** N_i. Also, as observed in Figure 2, noise variance σ_i^2 does not change linearly with ϵ_i. These altogether show that aggregation strategy $w_i \propto \epsilon_i$ is suboptimal. In contrast, Robust-HDP takes both of the sources of heterogeneity into account by assigning aggregation weights based on an estimation of $\{\sigma_i^2\}$ directly. With these settings, we got the results in Table 18 on CIFAR10, which shows superiority of Robust-HDP in this heterogeneity scenario.

G.2. Heterogeneous batch sizes $\{b_i\}$, uniform privacy parameters $\{\epsilon_i = \epsilon\}$ and heterogeneous dataset sizes $\{N_i\}$

In this section, we assume the same values for privacy parameters ($\epsilon_i = \epsilon$), but different batch and dataset sizes. Therefore, we have:

1. Effective clipping threshold:

$$\mathbb{E}[\tilde{g}_i(\boldsymbol{\theta})] = \frac{1}{b}\sum_{j\in\mathcal{B}_i^t}\mathbb{E}[\bar{g}_{ij}(\boldsymbol{\theta})] = \frac{1}{b}\sum_{j\in\mathcal{B}_i^t}G_i(\boldsymbol{\theta}) = G_i(\boldsymbol{\theta}), \tag{63}$$

$$\sigma_{i,\tilde{g}}^2 := \texttt{Var}(\tilde{g}_i(\boldsymbol{\theta})) = \frac{c^2 - \left\|G_i(\boldsymbol{\theta})\right\|^2}{b} + \frac{pc^2z^2(\epsilon,\delta,\frac{b_i}{N_i},K,E)}{b^2} \approx \frac{pc^2z^2(\epsilon,\delta,\frac{b_i}{N_i},K,E)}{b_i^2}. \tag{64}$$

2. Ineffective clipping threshold:

$$\mathbb{E}[\tilde{g}_i(\boldsymbol{\theta})] = \mathbb{E}[g_i(\boldsymbol{\theta})] = \nabla f_i(\boldsymbol{\theta}), \tag{65}$$

$$\sigma_{i,\tilde{g}}^2 = \texttt{Var}(\tilde{g}_i(\boldsymbol{\theta})) = \texttt{Var}(g_i(\boldsymbol{\theta})) + \frac{p\sigma_{i,\text{DP}}^2}{b^2} \leq \sigma_{i,g}^2 + \frac{pc^2z^2(\epsilon,\delta,\frac{b_i}{N_i},K,E)}{b_i^2}, \tag{66}$$

and

$$\sigma_i^2 := \texttt{Var}(\Delta\tilde{\boldsymbol{\theta}}_i^e|\boldsymbol{\theta}^e) = K\times\lceil\frac{1}{q_i}\rceil\cdot\eta_l^2\cdot\sigma_{i,\tilde{g}}^2 \approx K\cdot\frac{N_i}{b_i}\cdot\eta_l^2\cdot\sigma_{i,\tilde{g}}^2. \tag{67}$$

Hence, σ_i^2 varies across clients as a function of both b_i and N_i and heavily depends on b_i (b_i appears with power 3). Despite this heterogeneity in the set $\{\sigma_i^2\}_{i=1}^n$, WeiAvg assigns the same aggregation weights to all clients, due to their privacy parameters being equal, which is clearly inefficient. In contrast, Robust-HDP estimates the values in $\{\sigma_i^2\}_{i=1}^n$ directly and assigns larger weights to clients with larger batch sizes. With these settings and the Dirichlet data allocation mentioned above, we got the results in Table 19, which shows superiority of Robust-HDP in this case as well. We have used the mean values of the distributions Dist1, Dist3, Dist5, Dist7 and Dist9 from Table 7 for ϵ, i.e., $\epsilon \in \{\texttt{2.6}, \texttt{2.0}, \texttt{1.1}, \texttt{0.6}, \texttt{0.35}\}$. Also, as before, we have fixd δ_i to $\texttt{1e} - \texttt{4}$.

Table 19. Comparison of different algorithms on CIFAR10 with heterogeneous batch sizes $\{b_i\}$, uniform privacy parameters $\{\epsilon_i = \epsilon\}$ and heterogeneous dataset sizes $\{N_i\}$. FedAvg ($E = 200$) achieves 73.55%. "minimum ϵ" algorithm is equivalent to DPFedAvg in this case.

alg \ distr	$\epsilon = 2.6$	$\epsilon = 2.0$	$\epsilon = 1.1$	$\epsilon = 0.6$	$\epsilon = 0.35$
WeiAvg and PFA (Liu et al., 2021a)	35.86	33.50	29.21	**24.49**	18.40
DPFedAvg (Noble et al., 2021)	37.00	32.89	29.32	23.06	19.14
Robust-HDP	**37.45**	**34.93**	**29.78**	23.15	**19.54**

G.3. uniform batch sizes $\{b_i = b\}$, uniform privacy parameters $\{\epsilon_i = \epsilon\}$ and heterogeneous dataset sizes $\{N_i\}$

In this section, other than using the same values for clients batch sizes ($b_i = b$), we fix the privacy parameter of all clients to the same value ϵ (i.e., we have homogeneous DPFL, for which DPFedAvg has been proposed). Therefore, we have:

1. Effective clipping threshold:

$$\mathbb{E}[\tilde{g}_i(\boldsymbol{\theta})] = \frac{1}{b}\sum_{j\in\mathcal{B}_i^t}\mathbb{E}[\bar{g}_{ij}(\boldsymbol{\theta})] = \frac{1}{b}\sum_{j\in\mathcal{B}_i^t} G_i(\boldsymbol{\theta}) = G_i(\boldsymbol{\theta}), \tag{68}$$

$$\sigma_{i,\tilde{g}}^2 := \texttt{Var}(\tilde{g}_i(\boldsymbol{\theta})) = \frac{c^2 - \left\|G_i(\boldsymbol{\theta})\right\|^2}{b} + \frac{pc^2z^2(\epsilon,\delta,\frac{b}{N_i},K,E)}{b^2} \approx \frac{pc^2z^2(\epsilon,\delta,\frac{b}{N_i},K,E)}{b^2}. \tag{69}$$

2. Ineffective clipping threshold:

$$\mathbb{E}[\tilde{g}_i(\boldsymbol{\theta})] = \mathbb{E}[g_i(\boldsymbol{\theta})] = \nabla f_i(\boldsymbol{\theta}), \tag{70}$$

$$\sigma_{i,\tilde{g}}^2 = \texttt{Var}(\tilde{g}_i(\boldsymbol{\theta})) = \texttt{Var}(g_i(\boldsymbol{\theta})) + \frac{p\sigma_{i,\texttt{DP}}^2}{b^2} \leq \sigma_{i,g}^2 + \frac{pc^2z^2(\epsilon,\delta,\frac{b}{N_i},K,E)}{b^2}, \tag{71}$$

and

$$\sigma_i^2 := \texttt{Var}(\Delta\tilde{\boldsymbol{\theta}}_i^e|\boldsymbol{\theta}^e) = K\cdot\lceil\frac{1}{q_i}\rceil\cdot\eta_l^2\cdot\sigma_{i,\tilde{g}}^2 \approx K\cdot\frac{N_i}{b}\cdot\eta_l^2\cdot\sigma_{i,\tilde{g}}^2. \tag{72}$$

Hence σ_i^2 varies across clients as a function of only N_i. In the next paragraph, we show that this variation with N_i is small. This means that when clients hold the same privacy parameter and also use the same batch size, the amount of noise in their model updates sent to the server are almost the same, i.e., $\sigma_i^2 \approx \sigma_j^2, i \neq j$. **Hence, in this case the problem in Equation (8) has solution** $w_i \approx \frac{1}{n}$. In the following, we show what is the difference between the solutions provided by different algorithms for this case.

G.3.1. Performance parity in DPFL systems

Before proceeding to the experimental results, we draw your attention to the weight assignments by Robust-HDP in this setting, where both privacy parameters and batch sizes are uniform. Robust-HDP aims at approximating $\{\sigma_i^2\}$ and:

$$w_i^* \propto \frac{1}{\sigma_i^2} \approx \frac{b}{K\eta_l^2}\cdot\frac{1}{N_i\sigma_{i,\tilde{g}}^2} \approx \frac{b^3}{Kpc^2\eta_l^2}\cdot\frac{1}{N_iz^2(\epsilon,\delta,\frac{b}{N_i},K,E)} = \frac{b^3}{Kpc^2\eta_l^2}\cdot\frac{1}{H(N_i,b,\epsilon,\delta,K,E)} \tag{73}$$

where we have used Equation (3) (with b and ϵ) and $H(N_i,b,\epsilon,\delta,K,E) := N_iz^2(\epsilon,\delta,\frac{b}{N_i},K,E)$. Now note that z decreases with N_i sublinearly (see Figure 8. Remember that $q_i = \frac{b_i}{N_i}$). We have plotted the behavior of the function

Table 20. Comparison of different algorithms on CIFAR10 with uniform batch sizes $\{b_i = 32\}$, uniform privacy parameters $\{\epsilon_i = \epsilon\}$ and heterogeneous dataset sizes $\{N_i\}$. FedAvg ($E = 200$) achieves 73.55%. ”minimum ϵ” algorithm is equivalent to DPFedAvg in this case.

distr alg	$\epsilon = 2.6$	$\epsilon = 2.0$	$\epsilon = 1.1$	$\epsilon = 0.6$	$\epsilon = 0.35$
WeiAvg and PFA (Liu et al., 2021a)	37.24	**34.90**	27.80	23.22	19.01
DPFedAvg (Noble et al., 2021)	**38.52**	32.78	**30.42**	**23.50**	20.26
Robust-HDP	37.68	32.54	27.51	22.58	**20.52**

$H(N_i, b, \epsilon, \delta, K, E)$ as a function of N_i in Figure 11. Hence, *when N_i decreases, w_i^* increases slowly*. This means that Robust-HDP tries to minimize the noise in the aggregated model parameter (problem 8) and also assigns slightly larger weights to the clients with smaller datasets. Similarly, WeiAvg assigns uniform weights to all clients. In contrast, the solution provided by DPFedAvg focuses more on clients with larger train sets ($w_i \propto N_i$). Considering the point that $\{\sigma_i^2\}$ is almost uniform, this way it exploits clients with larger train sets during training. In the following, we discuss how this is related to performance fairness across clients.

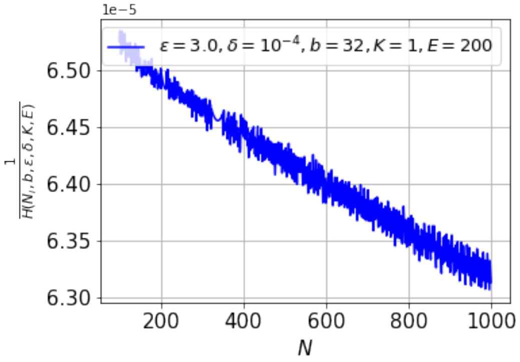

Figure 11. The behavior of function $1/H(N, b, \epsilon, \delta, K, E)$ as a function of N (dataset size) for an instance value of $\epsilon = 3$ and batch size $b = 32$. Oscillations in the curves are due to finding z empirically.

There have been multiple works in the literature , showing that DP has adverse effects on fairness in ML systems making it impossible to achieve both fairness and DP simultaneously (Cummings et al., 2019; Fioretto et al., 2022; Matzken et al., 2023). The work in (Bagdasaryan & Shmatikov, 2019) showed that accuracy of DP models drops much more for the underrepresented classes and subgroups, which yields to fairness issues. Interestingly, our Robust-HDP takes care of clients with minority data (i.e., those with small N_i) by assigning slightly larger weights to them at aggregation time, as shown in Equation (73) and Figure 11. Similarly, WeiAvg assigns uniform weights to clients. Hence, when both batch size and privacy parameters are uniform across clients (i.e., homogeneous DPFL), we expect the weight assignments of Robust-HDP and WeiAvg to yield to a higher performance fairness across clients, while we expect DPFedAvg - which was designed for homogeneous DPFL - to improve system utility. Our experimental results further clarifies this. We use the mean values of the distributions Dist1, Dist3, Dist5, Dist7 and Dist9 from Table 7 for ϵ, i.e., $\epsilon \in \{2.6, 2.0, 1.1, 0.6, 0.35\}$. Also, as before, we fix δ_i to $1\text{e} - 4$. With these settings and using the Dirichlet data allocation mentioned before, we got the results in Table 20 and Table 21.

G.4. Conclusion: when to use Robust-HDP?

We now summarize our understandings from the theories and experimental results in previous sections to conclude when to use Robust-HDP in DPFL settings. From our experimental results, heterogeneity in either of the privacy parameters $\{\epsilon_i\}_{i=1}^n$ and batch sizes $\{b_i\}_{i=1}^n$, results in a considerable heterogeneity in noise variances $\{\sigma_i^2\}_{i=1}^n$. Hence, using Robust-HDP in this cases will be beneficial in the system overall utility. However, if both $\{\epsilon_i\}_{i=1}^n$ and $\{b_i\}_{i=1}^n$ are homogeneous, and the only potential heterogeneity is in $\{N_i\}_{i=1}^n$ (i.e., homogeneous DPFL), then using DPFedAvg will be slightly better in terms of the system overall utility, as it assigns larger weights to the clients with larger dataset sizes. Despite this, using

Table 21. Comparison of different algorithms on CIFAR10 with uniform batch sizes $\{b_i = 32\}$, uniform privacy parameters $\{\epsilon_i = \epsilon\}$ and heterogeneous dataset sizes $\{N_i\}$ in terms of the std of clients' test accuracies and test accuracy of the client with the smallest train set (in parentheses). ”minimum ϵ” algorithm is equivalent to DPFedAvg in this case.

distr alg	$\epsilon = 2.6$	$\epsilon = 2.0$	$\epsilon = 1.1$	$\epsilon = 0.6$	$\epsilon = 0.35$
WeiAvg and PFA (Liu et al., 2021a)	**4.24** (32.95)	**4.11** (30.68)	4.95 (25.01)	**3.89** (**27.84**)	5.70 (17.04)
DPFedAvg (Noble et al., 2021)	4.92 (**34.69**)	4.71 (29.54)	4.95 (25.41)	6.78 (14.77)	6.86 (11.36)
Robust-HDP	4.77 (34.09)	4.59 (**34.91**)	**4.66** (**26.13**)	6.01 (15.34)	**3.89** (**18.97**)

Robust-HDP will alightly improve the performance of clients with smaller dataset sizes.

G.5. Gradient accumulation

When training large models with DPSGD, increasing the batch size results in memory exploding during training or finetuning. This might happen even when we are not using DP training. On the other hand, using a small batch size results in larger stochastic noise in batch gradients. Also, in the case of DP training, using a small batch size results in fast increment of DP noise (as explained in 3.2 in details). Therefore, if the memory budget of devices allow, we prefer to avoid using small batch sizes. But what if there is a limited memory budget? A solution for virtually increasing batch size is "gradient accumulation", which is very useful when the available physical memory of GPU is insufficient to accommodate the desired batch size. In gradient accumulation, gradients are computed for smaller batch sizes and summed over multiple batches, instead of updating model parameters after computing each batch gradient. When the accumulated gradients reach the target logical batch size, the model weights are updated with the accumulated batch gradients. The page in https://opacus.ai/api/batch_memory_manager.html shows the implementation of gradient accumulation for DP training.

H. Limitations and Future works

In this section, we investigate the potential limitations of our proposed Robust-HDP, and look at the future directions for addressing them. As before, we assume full participation of clients for simplicity. Specifically, we are curious about what happens if the data distribution across clients is not completely *i.i.d*, but rather is moderately/highly heterogeneous. We investigate Robust-HDP in these two scenarios in Appendix H.1 and Appendix H.2, respectively.

H.1. Robust-HDP with moderately heterogeneous data distribution

In order to evaluate Robust-HDP when the data split is moderately heterogeneous, we run experiments on MNIST. In order to simulate a controlled higher data heterogeneity, we use the sharding data splitting method described in Appendix B.1 and Table 6, *and we let each client to hold data samples of at maximum 8 classes*, with 60 clients in total. We consider two cases:

All 60 clients use the same batch size 128: the results obtained for this case, i.e., heterogeneous data with uniform batch sizes 128, were reported in Table 4. As we observed, Robust-HDP still outperforms the baselines in most of the cases. However, compared to the results in Table 12, which were obtained when the data split was *i.i.d*, its superiority has decreased. In order to get an understanding why this is the case, lets have a look at the aggregation weight assignments by different algorithms for this setting in Figure 12. Remember that, we have assumed uniform batch size of 128 for all clients. Therefore, the only parameter that makes variation in $\{\sigma_i^2\}$ is the clients' privacy parameters $\{\epsilon_i\}$ being different. There are multiple points in Figure 12. First, the accuracy of RPCA decomposition in estimating $\{\sigma_i^2\}$ has decreased (compare the difference between $\{\sigma_i^2\}$ and their estimates in Figure 12 with that in Figure 10 which was on a *i.i.d* data split). Second, despite this, the aggregation weights returned by Robust-HDP are very close to the optimum weights. This is the case because, as explained in Section 3.5, estimating the noise variances $\{\sigma_i^2\}$ up to a multiplicative factor suffices for Robust-HDP to get to the optimum aggregation weights $\{w_i^*\}$. Lastly, compared to the aggregation weights returned by WeiAvg, Robust-HDP has smoothly assigned larger weights to the clients with larger privacy parameters $\{\epsilon_i\}$. Note that, as we have assumed uniform batch size for all clients, having a larger privacy parameter ϵ is equivalent to having a less noisy model update sent to the server.

Table 22. Comparison of different algorithms (on MNIST, $E = 200$) with heterogeneous data split (maximum 8 labels per client) and 60 clients in the system with heterogeneous batch sizes.

distr alg	Dist1	Dist2	Dist3	Dist4	Dist5	Dist6	Dist7	Dist8	Dist9
WeiAvg (Liu et al., 2021a)	**84.60**	78.00	**83.70**	**78.42**	77.3	**75.23**	78.01	66.10.	68.12
PFA (Liu et al., 2021a)	79.07	75.72	83.23	76.54	78.06	64.00	79.10	**68.08**	71.74
DPFedAvg (Noble et al., 2021)	83.41	74.95	82.69	70.85	77.45	74.32	74.92	62.07	67.25
Robust-HDP	83.65	**79.38**	82.88	77.13	**83.53**	71.14	**80.41**	61.75	**71.76**

From the points mentioned above and the results in Figure 12, we conclude that, despite the moderate data heterogeneity, Robust-HDP *is still successful in assigning the aggregation wights* $\{w_i\}_{i=1}^n$ *such that the noise the aggregated model update is minimized.* But considering the heterogeneity in clients' data, is this good for the accuracy of the model too? More specifically, with data heterogeneity, does assigning larger weights to the clients with less noisy model updates necessarily result in higher utility too? From the results in Table 4, we observe that when the data is slightly heterogeneous and batch sizes are uniform, this is the case most of the times. However, as we will show next, this seems to be not the case when we also consider an additional heterogeneity in clients' batch sizes.

The batch sizes of the 60 clients are randomly selected from $\{\mathbf{16}, \mathbf{32}, \mathbf{64}, \mathbf{128}\}$: we recall Equation (7), which showed the considerable effect of batch size of a client on the noise variance in its model updates. When batch size decreases, its noise variance increases fast. Hence, unlike the previous case with uniform batch sizes, it is now both the batch sizes and privacy parameters of clients that determine the noise variance in their model updates. The results in Table 22 are obtained in this case. Also, Figure 13 compares the weight assignments by different algorithms.

As observed, Robust-HDP no longer outperforms the baselines. To get a better understanding, lets have a look at the aggregation weight assignments by different algorithms for this setting in Figure 13. First, the accuracy of RPCA decomposition in estimating $\{\sigma_i^2\}$ has again decreased compared to that in Figure 10, which was on a *i.i.d* data split. Second, despite this, the aggregation weights returned by Robust-HDP are still close to the optimum weights. However, the plot of assigned weights by Robust-HDP are more spiky than that in Figure 12: compared to the aggregation weights returned by WeiAvg, Robust-HDP has assigned larger weights to the clients with larger privacy parameters $\{\epsilon_i\}$ *and larger batch sizes*. Batch size of clients has a larger effect on the aggregation weights assigned to them. For instance client 59, which has batch size 128 and the second largest privacy parameter, has been assigned aggregation weight close to 0.18, while the same client got aggregation weight close to 0.05 when all clients used the same batch size (Figure 12). There are 6 clients whose aggregation weights sum to more than 0.5, i.e., these only 6 clients contribute to the aggregated model parameter more than the other 54 clients altogether. The reason behind this is that Robust-HDP aims at minimizing the noise level in the aggregated model update at the end of each round, and it has been successful in that. But the question is that, in this scenario with data heterogeneity, is this strategy beneficial for the utility of the final trained model too? Although, this strategy results in maximizing the trained model utility when the data split is *i.i.d*, it is not the case when we have data heterogeneity and batch size heterogeneity simultaneously, and the results in Table 22 confirm this. This is a limitation for Robust-HDP. However, we can provide a solution for it. Heterogeneity in batch sizes usually happens when clients have different memory budgets. Clients with low memory budgets can not use large batches, especially when training privately with DPSGD (Abadi et al., 2016). As observed, when having data heterogeneity, this batch size heterogeneity deteriorates the performance of Robust-HDP. Despite the heterogeneity that may exist in clients' memory budgets, they can use gradient accumulation explained in Appendix G.5 to virtually increase their batch sizes to a uniform batch size (e.g., 128). In this case, we get back to the results in Table 4, in which Robust-HDP works well most of the times. The cost that we pay is that clients with limited physical memory sizes have to spend more time locally during each global round, and the server should wait longer for these clients before performing each aggregation.

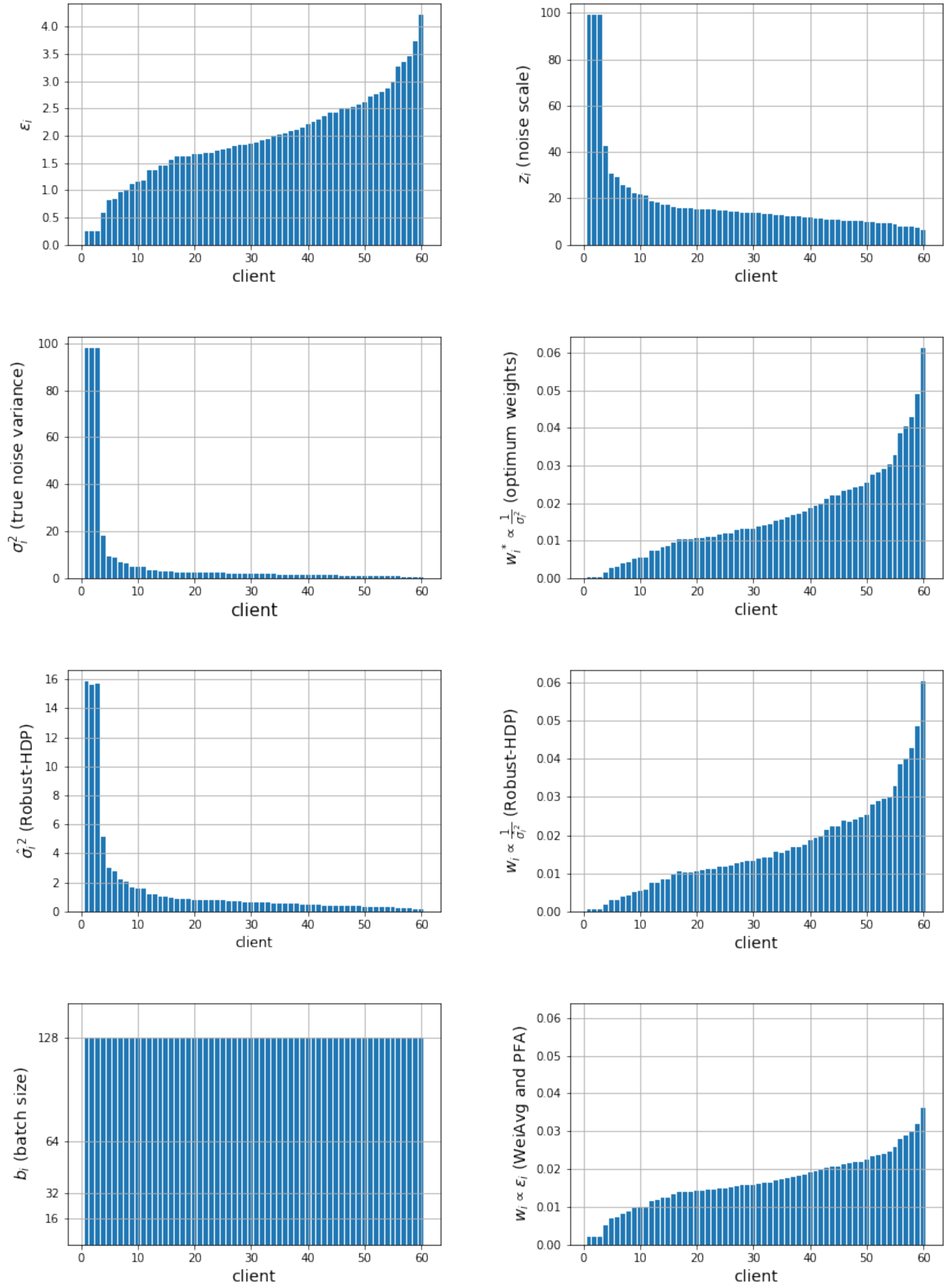

Figure 12. Comparison of weight assignments by different algorithms on MNIST, Dist1 and 60 clients with heterogeneous data distribution (maximum 8 labels per client) and uniform batch size ($b_i = 128$). The weight assignments by Robust-HDP are very close to the optimal weight assignment strategy, despite the heterogeneity in the data split. Also, the first 40 (last 20) clients, which have the smallest (largest) ϵ privacy parameters, get assigned smaller (larger) weights by Robust-HDP than by PFA and WeiAvg, **showing the suboptimality of aggregation strategy of** PFA **and** WeiAvg. The results for this experiment were reported in Table 4 (Robust-HDP, Dist1).

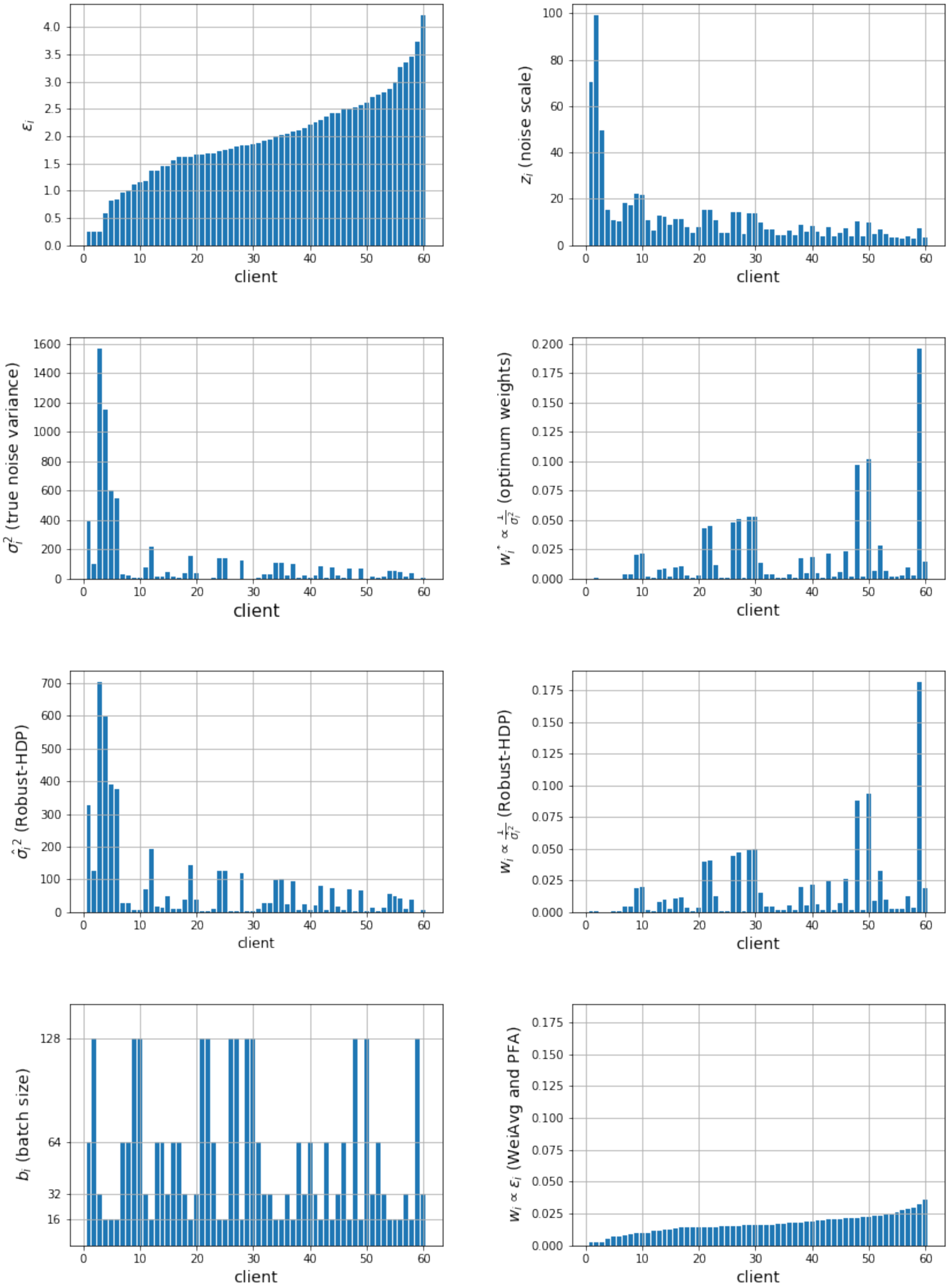

Figure 13. Comparison of weight assignments by different algorithms on MNIST, Dist1 and 60 clients with hetrogeneous data distribution (maximum 8 labels per client) and heterogeneous batch sizes. The weight assignments by Robust-HDP are close to the optimal weight assignment strategy, despite the heterogeneity in the data split. The results for this experiment were reported in Table 22 (Robust-HDP, Dist1).

H.2. Local DPFL with highly heterogeneous data split across clients (future work)

Having studied Robust-HDP in scenarios with *i.i.d* and slightly heterogeneous data splits, we are curious about the scenarios with highly heterogeneous data splits. In non-private FL systems, high data heterogeneity is usually addressed by personalized FL (Li et al., 2020) and clustered FL (Sattler et al., 2019). In the former case, each client learns a model specifically for itself by fine-tuning the common model obtained from FL on its local data. In the latter case, clients with similar data are first grouped into a cluster by the server, followed by federated training of a model for each cluster. In highly heterogeneous data distributions, clustered FL is more common (Sattler et al., 2019; Werner et al., 2023).

On the other hand, we have DPFL systems with highly heterogeneous data splits. In the existence of a trusted server (CDP), an idea was proposed by Chathoth et al. (2022) for clustering clients with cohort-level privacy with privacy and data heterogeneity across cohorts, using ϵ-DP definition (Definition 2.1 with $\delta = 0$). When there is no trusted server (LDP), we can follow a similar direction of clustered DPFL to address scenarios with highly heterogeneous data splits: clients are first clustered by the server such that the data distribution of clients in a cluster are more similar to each other, and then, a model is learned for each cluster. However, the DP noise in clients' model updates makes clustering of clients harder. A recent work in (Malekmohammadi et al., 2024) has addressed this scenario by proposing an algorithm, which is robust to the DP noise existing in clients' model updates, for clustering clients in DPFL system with local differential privacy.