Fast, Private and Fair Federated Learning

by

Saber Malekmohammadi

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2025

© Saber Malekmohammadi 2025

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner: Shahab Asoodeh
Assistant Professor
Department of Computing and Software, McMaster University

Supervisor: Yaoliang Yu
Associate Professor
School of Computer Science, University of Waterloo

Internal Members: Gautam Kamath
Assistant Professor
School of Computer Science, University of Waterloo

Hong Zhang
Assistant Professor
School of Computer Science, University of Waterloo

Internal-External Member: Chengguo Weng
Professor
Department of Statistics and Actuarial Science, University of Waterloo

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

This thesis includes a presentation of my research works, summarized from three papers, including one conference paper, one book chapter and one paper under review. I am the first author of all the three papers, where I mainly contributed ideas, implemented the proposed methods, and wrote the papers. My co-authors are my supervisor and my collaborators, who provided me with invaluable help in refining ideas and polishing my writings.

- Malekmohammadi et al. 2021a,b, 2023: “A Unifying Framework for Federated Learning". In Springer Book on Federated and Transfer Learning, which corresponds to chapter 2.
- Malekmohammadi et al. 2024: “Noise-Aware Algorithm for Heterogeneous Differentially Private Federated Learning". In International Conference on Machine Learning (ICML), which corresponds to chapter 3.
- Malekmohammadi et al. 2025: “Differentially Private Clustered Federated Learning". On ArXiv and under the review of NeurIPS 2025, which corresponds to chapter 4.

Abstract

A remarkable progress in machine learning (ML) technologies has happened during the past decade. Federated learning (FL) is a decentralized ML setting, where some clients (data holders) collaborate with each other to train a model under the orchestration of a central server. Clients, for instance, can be some mobile devices (in cross-device FL) or some organizations (in cross-silo FL). In FL, the training data of clients remain decentralized, mitigating many systemic privacy risks and costs existing in centralized ML. This thesis investigates three main sub-problems of FL, including: 1. Convergence analysis of FL algorithms 2. Formal data privacy guarantees for clients through differentially private FL (DPFL) 3. Performance parity in DPFL systems under structured data heterogeneity. We provide further details about them in the following.

Over the past years, the FL community witnessed a plethora of new algorithms. However, there is a lack of thorough comparison of these algorithms and the theory behind them. Our fragmented understanding of the theory behind the existing algorithms is the reason behind the lack of such a formal comparison. This is the main focus of Chapter 2, where we show that many of the existing FL algorithms can be understood from the lens of operator splitting theory. This unification allows us to compare different algorithms easily, to refine previous convergence results and to propose new algorithmic variants. Our new theoretical findings reveal the remarkable role played by the step size in FL. Furthermore, the unification allows us to propose a simple and economic way to accelerate the convergence of different algorithms. Fast convergence is vital in FL, as communication between the server and clients is often considered as an overhead.

Despite the fact that FL provides clients with some level of informal data privacy by operating on their decentralized data, the orchestrating server or other malicious third parties can still attack the data privacy of participating clients. Consequently, FL has been augmented with differential privacy (DP) in order to provide rigorous data privacy guarantees to clients. In DPFL, which is the focus of Chapter 3, there is often heterogeneity in the privacy requirements of clients due to their varying privacy policies. This heterogeneity as well as the heterogeneity in batch and/or dataset sizes of clients lead to a variation in the DP noise level across clients' model updates. Hence, straightforward aggregation strategies, e.g. assigning clients' aggregation weights proportional to their privacy parameters, will lead to a low utility for the system. We propose a DPFL algorithm which efficiently estimates the true noise level in clients' uploaded model updates and uses an aggregation strategy to reduce the noise level in their aggregation. Our proposed method improves utility and convergence speed, while being safe to the clients that may maliciously send falsified privacy parameters to the server to attack the system's utility.

Finally, in Chapter 4, we investigate the intersection of FL, DP and performance fairness, under structured data heterogeneity across clients. This type of data heterogeneity is often addressed through clustering clients (a.k.a clustered FL). However, the existing clustered FL methods remain sensitive and prone to errors, further exacerbated by the DP noise in DPFL systems. In this chapter, we propose a robust algorithm for differentially private clustered FL. To this end and justified by our theoretical results, we propose to use large batch sizes in the first round and cluster clients based on both their model updates as well as their training loss values. Furthermore, for

clustering clients' model updates at the end of the first round, our proposed approach addresses the server's uncertainties by employing Gaussian mixture models (GMM) to reduce the impact of DP and stochastic noise and avoid potential clustering errors. This idea is efficient especially in privacy-sensitive scenarios with more DP noise and leads to a high accuracy for clustering clients.

Acknowledgements

Hereby, I express my gratitude to my PhD supervisor, Yaoliang, who provided me with valuable research advice. I also thank him for his support, patience and guidance.

I also would like to thank the examining committee; Prof. Sh. Asoodeh, Prof. Ch. Weng, Prof. G. Kamath and Prof. H. Zhang for their valuable time.

The Cheriton School of Computer Science staff, especially Paula Roser and Nadine Zinger (the previous and current CS PhD advisors) were always helpful and supportive. I thank them for their help.

Finally, I would like to thank my family, especially my mother, for their constant support.

Dedication

To my family.

Table of Contents

Examining Committee ii

Author's Declaration iii

Statement of Contributions iv

Abstract v

Acknowledgements vii

Dedication viii

List of Figures xiv

List of Tables xxi

1 Introduction 1

1.1 Federated learning . . . 2

1.1.1 FL with one global model . . . 2

1.1.2 Personalized FL . . . 2

1.1.3 Clustered FL . . . 3

1.1.4 Performance fairness in FL . . . 3

1.2 Differential privacy (DP) . . . 4

1.2.1 ϵ-DP . . . 4

1.2.2 Approximate $(\epsilon, \delta)-$DP . . . 5

1.2.3 Rényi DP . . . 5

1.2.4 Zero concentrated DP (zCDP) . . . 6
1.2.5 Exponential mechanism for private selection . . . 6
1.2.6 Differentially private SGD (DPSGD) . . . 7
1.3 Outline and contributions . . . 7

2 An Operator Splitting View of Federated Learning 9
2.1 Preface . . . 9
2.2 Introduction . . . 9
2.3 Background . . . 11
2.4 Federated learning as operator splitting . . . 12
2.4.1 FedAvg as forward-backward splitting . . . 12
2.4.2 FedProx as backward-backward splitting . . . 15
2.4.3 FedSplit as Peaceman-Rachford splitting . . . 18
2.4.4 FedPi as Douglas-Rachford splitting . . . 18
2.4.5 FedRP as Reflection-Projection splitting . . . 20
2.4.6 Effect of data heterogeneity . . . 21
2.5 Unification, implementation and acceleration . . . 21
2.6 Summary . . . 24

3 Noise-Aware Algorithm for Heterogeneous Differentially Private Federated Learning 25
3.1 Preface . . . 25
3.2 Introduction . . . 25
3.3 Related work . . . 26
3.4 Notations . . . 28
3.5 Robust-HDP algorithm for heterogeneous DPFL . . . 28
3.5.1 Noise level in clients' DP batch gradients . . . 30
3.5.2 Noise level in clients' DP model updates . . . 30
3.5.3 Optimum aggregation strategy . . . 31
3.5.4 Robust-HDP algorithm . . . 32
3.5.5 Scalability of Robust-HDP with the number of model parameters p . . . 33
3.5.6 Reliability of Robust-HDP . . . 34

3.5.7 Privacy analysis of Robust-HDP . . . 34
3.5.8 The optimization side of Robust-HDP . . . 35
3.6 Experiments . . . 36
3.6.1 Experimental setup . . . 36
3.6.2 Experimental results . . . 37
3.7 Summary . . . 41

4 A Provably Robust Algorithm for Differentially Private Clustered Federated Learning 42
4.1 Preface . . . 42
4.2 Introduction . . . 42
4.3 Related work . . . 44
4.4 Definitions, notations and assumptions . . . 45
4.5 Methodology and proposed algorithm . . . 47
4.5.1 R-DPCFL algorithm . . . 48
4.5.2 Reducing GMM uncertainty via using full batch sizes in the first round and small batch sizes in the subsequent rounds . . . 50
4.5.3 Effect of batch sizes $\{b_i^1\}_{i=1}^n$ on the separation between clusters . . . 52
4.5.4 Convergence rate of EM for learning GMM . . . 52
4.5.5 Applicability of R-DPCFL . . . 53
4.6 Experiments . . . 53
4.6.1 Baseline selection . . . 54
4.6.2 Results . . . 54
4.7 Summary . . . 57

5 Conclusion 58

References 68

A Appendix of Chapter 2 69
A Experimental Setup . . . 69
A.1 Experimental setup: least squares and logistic regression . . . 69
A.2 Experimental setup: MNIST datasets . . . 70

A.3 Experimental setup: CIFAR-10 dataset . . . 70
B Additional experimental results . . . 72
B.1 Effect of client sampling . . . 72
B.2 Effect of using mini-batch gradient descent locally . . . 74
B.3 Results with error bars for Figure 2.8, Right . . . 74
C Proofs . . . 76

B Appendix of Chapter 3 **81**
A Experimental setup . . . 81
A.1 Datasets and models . . . 81
A.2 DP training parameters . . . 82
A.3 Algorithms to compare and tuning hyperparameters . . . 82
B Derivations . . . 84
C Used lemmas . . . 86
D Proofs . . . 87
D.1 Proof of Theorem 9 . . . 87
D.2 Proof of Theorem 10 . . . 87
E Detailed results . . . 97
E.1 Test accuracy comparison . . . 97
E.2 Ablation study on privacy level and number of clients . . . 97
E.3 Precision of Robust-HDP . . . 98
F Additional Experiments . . . 102
F.1 Uniform batch sizes $\{b_i = b\}$, heterogeneous privacy parameters $\{\epsilon_i\}$ and heterogeneous dataset sizes $\{N_i\}$ . . . 102
F.2 Heterogeneous batch sizes $\{b_i\}$, uniform privacy parameters $\{\epsilon_i = \epsilon\}$ and heterogeneous dataset sizes $\{N_i\}$ . . . 104
F.3 Uniform batch sizes $\{b_i = b\}$, uniform privacy parameters $\{\epsilon_i = \epsilon\}$ and heterogeneous dataset sizes $\{N_i\}$ . . . 104
F.4 Conclusion: when to use Robust-HDP? . . . 107
G Gradient accumulation . . . 108
H Limitations and future works . . . 108
H.1 Robust-HDP with moderately heterogeneous data distribution . . . 108
H.2 DPFL with highly heterogeneous data split across clients (future work) . . . 110

C Appendix of Chapter 4 **113**

A Privacy accounting and budgeting . . . 113

B Experimental setup . . . 114

B.1 Datasets . . . 114

B.2 Models and optimization . . . 114

B.3 MR-MTL formulation . . . 115

B.4 Tuning hyperparameters of baseline algorithms . . . 115

B.5 Implementation of private local clustering for IFCA and R-DPCFL . . . 116

B.6 DP privacy parameters . . . 117

B.7 Gaussian mixture model . . . 117

C Setting hyper-parameters of R-DPCFL . . . 118

C.1 Batch size $b_i^{>1}$ . . . 118

C.2 The strategy switching time E_c . . . 118

C.3 The number of clusters M . . . 120

D Complete experimental results . . . 122

E Proofs . . . 123

E.1 Proof of Lemma 4 . . . 123

E.2 Proof of Lemma 5 . . . 126

E.3 Proof of Theorem 11 . . . 130

E.4 Formal privacy guarantees of R-DPCFL . . . 130

F The relation between Lemma 4 and the law of large numbers . . . 131

G Further related works . . . 131

List of Figures

2.1 When $f = \iota_C$ is the indicator function of a (closed) set C, $\mathsf{P}_f^\eta(\mathbf{w})$ is the Euclidean projection of $\mathbf{w}$ onto the set C. Similarly, $\mathsf{R}_f^\eta(\mathbf{v})$ is the Euclidean reflection of $\mathbf{v}$ w.r.t. the set C. 12

2.2 Convergence of FedAvg when changing k or η. Top: least squares; Bottom: logistic regression. 14

2.3 Running FedAvg on CIFAR-10 with different local epochs k. The number of clients is 10, local learning rate $\eta = 0.1$. Data points are sampled and distributed to clients non-uniformly. 14

2.4 Optimality gap $\{f(\boldsymbol{\theta}^*_{\mathsf{FedAvg}}) - f^*\}$ or training loss $\{f(\boldsymbol{\theta}^*_{\mathsf{FedAvg}})\}$ of (approximate) fixed-point solutions of FedAvg for different learning rates η and local epochs k. Different colored lines are for different numbers of local epochs, and dashed lines for different product values $\eta(k-1)$. Left: least squares (closed-form solution); Middle: logistic regression (6000 communication rounds); Right: nonconvex CNN on the MNIST dataset (200 communication rounds). 15

2.5 Effect of step size η and ergodic averaging on FedProx . Left: least squares; Middle: logistic regression; Right: CNN on MNIST. The dashed and solid lines with the same color show the results obtained with and without the ergodic averaging step in Theorem 3, respectively. For exponentially decaying η_t, we use period T equal to 500 for both least squares and logistic regression experiments, and 10 for CNN experiment. 18

2.6 Effect of step size η and averaging on FedRP . Left: least squares; Right: logistic regression. The dashed and solid lines with the same color show the results obtained with and without the ergodic averaging step in Theorem 3, respectively. For exponentially decaying η_t, we use period T equal to 500 for both least squares and logistic regression. 20

2.7 The effect of data heterogeneity on the performance of different splitting methods. The top row shows the results for the least squares, and the bottom row shows the results for nonconvex CNN model. Top-Left: small data heterogeneity with $H \approx 119 \times 10^3$; Top-Middle: moderately data heterogeneity with $H \approx 7.61 \times 10^6$; Top-Right: large data heterogeneity with $H \approx 190.3 \times 10^6$. Bottom-Left: i.i.d. data distribution; Bottom-Middle: non-i.i.d. data distribution with maximum 6 classes per client; Bottom-Right: non-i.i.d. data distribution with maximum 2 classes per client. 22

2.8 Effect of Anderson acceleration. Left: least squares with $\tau = 2$; Middle: logistic regression $\tau = 2$; Right: nonconvex CNN with $\tau = 10$. Dashed lines are the accelerated results. 24

3.1 Security model in heterogeneous DPFL, where client i has local train data $\mathcal{D}_i$ and record-level privacy parameters (ϵ_i, δ_i), and does not trust any external parties. . . 27

3.2 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). 31

3.3 **Left:** 3D plot of noise variance σ_i^2 (Equation (3.5) 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 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 corresponds to a point in the left figure. 32

3.4 Comparison of average test accuracy between studied algorithms. See Tables B.7 to B.10 in the Appendix for detailed results. 38

3.5 Convergence speed comparison on MNIST and Dist6. Minimum ϵ algorithm diverged in 1 out of 3 trials. 38

3.6 Performance comparison on MNIST. **Left:** effect of clients desired privacy on utility (detailed results in Table B.11 in the Appendix). **Middle:** effect of number of existing clients (privacy parameters of clients are sampled from Dist6) on utility (detailed results in Table B.12). **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. 39

3.7 **Left:** Precision of Robust-HDP (red) compared to oracle optimum strategy (blue) for CIFAR10 and Dist2, when using the approximation method in Section 3.5.5 with $Q = 1$ and $p' = 2 \times 10^5$. **Right:** Effect of the parameter p', used in the approximation method explained in Section 3.5.5, on the precision of the weights returned by Robust-HDP for CIFAR10 (with $p \approx 11 \times 10^6$). 40

4.1 Loss-based clustering algorithms miscluster in the initial rounds, due to model initialization. Also, even with the assumption of perfect clustering of clients in the first rounds, clustering algorithms based on gradients (model updates) leads to clustering errors in the last rounds, due to the gradients approaching to zero. . . . 46

4.2 **Left:** Considered threat model in this work, where client i has local train data $\mathcal{D}_i$ and "sample-level" DP privacy parameters (ϵ, δ), and does not trust any external party. **Right:** Three main stages of the proposed R-DPCFL algorithm. 48

4.3 Plot of $\texttt{Var}(\Delta\tilde{\boldsymbol{\theta}}_i^1(b_i^1)|\boldsymbol{\theta}_i^{init})$ **(left)** and $\texttt{Var}(\Delta\tilde{\boldsymbol{\theta}}_i^e(b_i^e)|\boldsymbol{\theta}_i^{e,0})$ $(e > 1)$ **(right)** v.s. both b_i^1 and $b_i^{>1}$. There are two clear takeaways: 1) for all $e \in \{1, \cdots, E\}$, $\texttt{Var}(\Delta\tilde{\boldsymbol{\theta}}_i^e(b_i^e)|\boldsymbol{\theta}_i^{e,0})$ decreases with b_i^e steeply (from Lemma 4). 2) The effect of $b_i^{>1}$ on $\texttt{Var}(\Delta\tilde{\boldsymbol{\theta}}_i^1(b_i^1)|\boldsymbol{\theta}_i^{init})$ (left figure) is considerable. The reason is that $b_i^{>1}$ is used in $E-1$ rounds and affects the noise scale $z_i(\epsilon, \delta, b_i^1, b_i^{>1}, N_i, K, E)$ used by DPSGD: **see Figure 4.4 for the plot of** $z_i(\epsilon, \delta, b_i^1, b_i^{>1}, N_i, K, E)$ **v.s.** b_i^1 **and** $b_i^{>1}$. The results are obtained on CIFAR10 from Rényi-DP Accountant (Mironov et al. 2019) in a setting with $N_i = 6600, \epsilon = 5, \delta = 10^{-4}, c = 3, K = 1, E = 200, p = 11,181,642, \eta_l = 5 \times 10^{-4}$. 50

4.4 Plot of $z_i(\epsilon, \delta, b_i^1, b_i^{>1}, N_i, K, E)$ v.s. b_i^1 and $b_i^{>1}$ obtained from Rényi-DP Accountant (Mironov et al. 2019) in a setting with $N_i = 6600, \epsilon = 5, \delta = 10^{-4}, K = 1, E = 200$. It is clearly observed that the effect of $b_i^{>1}$ is much more than the effect of b_i^1. The reason is that $b_i^{>1}$ is used in $E-1$ rounds, while b_i^1 is used only in the first round. So the value of $b_i^{>1}$ affects z_i the most. 51

4.5 PCA visualization of updates $\{\Delta\tilde{\boldsymbol{\theta}}_i^1\}_{i=1}^n$ on 2D space. **Left:** $\epsilon_i = 10$, $b_i^e = 32$ for all i and e. **Right:** $\epsilon_i = 10$, $b_i^1 = b^1 = N = 6600$, **i.e. full batch sizes** (assuming $N_i = N = 6600$ for all clients), and $b_i^{>1} = 32$ for all i. $\mu_m^*(b^1)$ shows the center of cluster m, when all clients use batch size b^1 during the first round. The model updates are obtained from clients running DPSGD for $K = 1$ epochs locally on CIFAR10 with covariate shift (rotation) across clusters, and under the same values as in Figure 4.3. 51

4.6 **Top:** Average test accuracy for different total privacy budgets ϵ. Results are from four different runs. 10% means performing local clustering by clients only in 10% of the total number of rounds; i.e. rounds $E_c \leq e \leq E_c + \lfloor \frac{E}{10} \rfloor$ for R-DPCFL and rounds $1 \leq e \leq 1 + \lfloor \frac{E}{10} \rfloor$ for IFCA (see Appendix B.5). **Figure D.3 in the Appendix includes the** Global **baseline too**. **Bottom:** Number of times (out of 4 runs) that R-DPCFL and IFCA successfully detect the underlying cluster structure of all existing clients. 55

4.7 **Top:** Average test accuracy across clients belonging to the minority cluster for different total privacy budgets ϵ, averaged over four different runs (random seeds). **Bottom:** Number of times (out of 4 runs) that R-DPCFL and IFCA successfully detect the minority cluster. 56

4.8 MSS score v.s. ϵ for two different local dataset sizes. A small local dataset size can be compensated for by using smaller batch sizes $\{b_i^{>1}\}_{i=1}^n$ to get a larger MSS score. 56

B.1 Effect of client sampling. Top-Left: least squares with $p = 0.5$ (participation probability); Top-Right: logistic regression with $p = 0.5$; Bottom-Left: least squares with $p = 0.7$; Bottom-Right: logistic regression with $p = 0.7$. 73

B.2 The effect of using minibatch gradient descent. Local datasets are divided into 5 minibatches. Top-Left: least squares with $B = 1000$. Top-Right: logistic regression with $B = 200$. Bottom: nonconvex CNN on MNIST with $B = 500$. . 75

B.3 Detailed results for Anderson acceleration for nonconvex optimization problem of CNN training with $\tau = 10$. Dashed lines are the accelerated results. 75

E.1 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 B.7 (Robust-HDP, Dist3 and Dist9). 100

E.2 Comparison of weight assignments for Dist8 and MNIST with the data split in Table B.2. 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 B.7 (Robust-HDP, Dist8). 101

F.3 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. 106

H.4 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 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. The results for this experiment were reported in Table 3.5 (Robust-HDP, Dist1). 111

H.5 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 B.17 (Robust-HDP, Dist1). 112

C.1 The effect of increasing the batch size after the first round, i.e. $b_i^{>1}$, on the model updates $\{\Delta\tilde{\boldsymbol{\theta}}_i^1\}_{i=1}^n$ at the end of the first round. All clients have used full batch sizes in the first round, i.e. $\forall i : b_i^1 = N_i$. The level of noise in $\{\Delta\tilde{\boldsymbol{\theta}}_i^1\}_{i=1}^n$ affects the quality and confidence of the client clustering that the server performs at the end of the first round. As can be observed, for a fixed $\epsilon = 10$, the model updates scatter further in space as $b_i^{>1}$ increases and different clusters get less separated. This leads to a decrement in the confidence level or the (MSS) score of the resulting GMM, as $b_i^{>1}$ increases (mentioned on the top of each plot). All the results are obtained on CIFAR10 with covariate shift (rotation) across clusters. . . 119

C.2 The minimum pairwise separation score (MSS) or confidence of the GMM learned on $\{\Delta\tilde{\boldsymbol{\theta}}_i^1\}_{i=1}^n$ peaks at the true cluster number, which is equal to 4 in all the plots above. Each figure is for a different value of ϵ (mentioned on top of each figure), and are obtained on CIFAR10 with covariate shift (rotation) across clusters, and 5 different random data splits (5 seeds). All the results are obtained with full batch sizes in the first round and $b_i^{>1} = 32$ for all i. We can use this observation as a method to find the true number of clusters (M) when it is not given. For larger ϵ, this method work perfectly and even when ϵ is too small, e.g., $\epsilon = 3$, this method works well and predicts the true number of clusters correctly most of the times: 3 out of the 5 curves in the bottom right plot have a peak at $M = 4$ (the true cluster number). and the other 2 curves predict 5 as the true number, which is the closest and the best alternative for the true value $M = 4$. 121

D.3 Average test accuracy across clients for different total privacy budgets ϵ (results are obtained from 4 different random seeds). 10% means performing loss-based clustering by clients only in 10% of the total rounds (E). 122

D.4 Average test accuracy across clients belonging to the minority cluster for different total privacy budgets ϵ (results are obtained from 4 different random seeds). 10% means performing loss-based clustering by clients only in 10% of the total rounds (E). 122

D.5 The MSS score of the learned GMM is indicative of whether the true underlying clusters will be detected or not: an MSS score above 2 always leads to correct detection of clusters. Each point is the result of one independent experiment. . . . 123

D.6 The number of EM iterations needed for learning the GMM decreases as b_i^1 increases. Especially, at full batch size ($\forall i : b_i^1 = N_i$, very few iterations of EM are needed. The results are obtained on CIFAR10 with covariate shift across $M = 4$ clusters. According to Lemma 5, using a large enough batch size b_i^1 in the first round makes the underlying clusters in $\{\Delta\tilde{\boldsymbol{\theta}}_i^1\}_{i=1}^n$ more distinguishable. Consequently, according to Theorem 11, convergence rate of EM algorithm for learning the GMM increases with b_i^1. 123

E.7 **Left:** Distance between the centers of different clusters, i.e. the distance between $\mu_m^*(b^1)$ and $\mu_{m'}^*(b^1)$, decreases k times as b^1 increases k times. The three curves in the plot are obtained on CIFAR10 with 4 clusters $m \in \{0, 1, 2, 3\}$ obtained from covariate shift (rotation). The curves are overlapping all with slope 0.95, which is very close to 1. This shows the validity of the approximation in Equation (C.8). **Right:** Effect of changing batch size b^1 to full batch size in the first round on the noise scale z. In the denominator, b^1 is equal to $b^{>1}$. Results are obtained from Rényi-DP accountant (Mironov et al. 2019) with $N = 50000$, $K = 1$ and $E = 200$. For each value of ϵ, we have shown the results for seven values of $b^{>1}$. 129

List of Tables

2.1 A unifying framework (2.15)-(2.17) for FL. Note that (a) FedAvg replaces the proximal update $\mathsf{P}_{\mathsf{f}}^{\eta_t}$ with a gradient update $\mathsf{G}_{\mathsf{f}}^{\eta_t}$; (b) ? indicates properties that remain to be studied; (c) "sampling" refers to selecting a subset of clients while "stochastic" refers to updating with stochastic gradient. 23

3.1 Features of different heterogeneous DPFL algorithms. $\times$: needed at server, ✓: not needed. 28

3.2 Notations used in Chapter 3 29

3.3 Distributions of privacy parameters (ϵ), from which clients' privacy parameters are sampled. 37

3.4 The average *per parameter* noise variance (Equation (3.5) and Equation (3.3)) normalized by used learning rate $(\frac{\sum_{i=1}^{n} w_i^{e\,2}\sigma_i^2}{p\eta_t^2})$ 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. . 39

3.5 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 $\mathbf{128}$. 40

4.1 Used notations in chapter 4. 46

A.1 MNIST model 71

A.2 CIFAR-10 model 71

B.1 CNN model for classification on MNIST/FMNIST datasets 82

B.2 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 B.1. 82

B.3 The learning rates used for training with each algorithm on MNIST dataset 84

B.4 The learning rates used for training with each algorithm on FMNIST dataset . . . 84
B.5 The learning rates used for training with each algorithm on CIFAR10 dataset . . . 84
B.6 The learning rates used for training with each algorithm on CIFAR100 dataset. . . 85
B.7 Comparison of different algorithms (on MNIST, $E = 200$). FedAvg achieves 98.6%. 97
B.8 Comparison of different algorithms (on FMNIST, $E = 200$). FedAvg achieves 90.28%. 97
B.9 Comparison of different algorithms (on CIFAR10, $E = 200$). FedAvg achieves 73.55%. 98
B.10 Comparison of different algorithms (on CIFAR100, $E = 200$). FedAvg achieves 61.80%. 98
B.11 Detailed results for ablation study on the privacy level of clients in Figure 3.6, left. 98
B.12 Detailed results for ablation study on number of clients in Figure 3.6, middle. . . 99
B.13 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. 103
B.14 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. 105
B.15 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. 107
B.16 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. 107
B.17 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. 110

C.1 CNN model for classification on MNIST/FMNIST datasets 114
C.2 Details of the used datasets in the main body of the paper. ResNet-18 is the residual neural networks defined in He et al. 2015. CNN: Convolutional Neural Network defined in Table B.1. 115

Chapter 1

Introduction

The fast progress in computational resources has enabled a rapid advancement in development of large deep learning (DL) models for various learning tasks. Central to this development is the need to data to train the models. There are various scenarios that the required data is naturally distributed among some data holders, e.g., among the clients in a communication network or the hospitals in a country. This scattered nature of data often comes from the privacy concerns of the holders, which does not let them to share their data with others or a central server. For instance, the clients in a communication network do not like to share the images stored in their devices with others. Also, it could also be expensive to transfer the data stored in different locations to a central server, e.g., the set of patients' medical records stored in different hospitals of a country.

Federated Learning (FL) (McMahan et al. 2017a) has been proposed as a decentralized machine learning (ML) framework for training ML/DL models under the orchestration of a central server on the data scattered between some devices/organizations, which we refer to as *clients*. The clients do not share their private data with neither other clients nor the central server. This is a beneficial feature of FL, which provides some level of informal data privacy to the participating clients. Nevertheless, there are some challenges for the deployment of FL frameworks in real-world applications. Among them are the expensive communication overhead between the server and the clients (McMahan et al. 2017a) as well as the potential data heterogeneity across the clients, which leads to disparate performance of the learned model on the participating clients' data (Li et al. 2020a,d; Zhang et al. 2023). Furthermore, in the presence of sensitive data in clients' datasets, they may need a rigorous data privacy guarantee in order to participate in FL (Geiping et al. 2020; Hitaj et al. 2017; Rigaki and García 2020; Wang et al. 2019; Zhu et al. 2019). The aforementioned challenges are vital to be addressed to ensure that FL systems can be reliably and effectively deployed.

The primary aim of this thesis is to address the above-mentioned challenges in FL, hoping that it will make the promising FL framework more applicable for real-world scenarios with distributed data. In the following sections of this chapter, we will present an introduction to various FL algorithms, performance parity in FL systems and also differential privacy. We will finally summarize the contributions of this thesis.

1.1 Federated learning

We start with a more detailed introduction to FL, its objective and some common FL algorithms.

1.1.1 FL with one global model

As mentioned earlier, an FL system consists of a central server, and some participating clients. Let n denote the number of participating clients, where client i has the loss function $f_i(\boldsymbol{\theta})$, parameterized by $\boldsymbol{\theta}$, on its local dataset $\mathcal{D}_i$. Then, the problem of FL is about collectively and efficiently optimizing the individual objective functions f_i in a decentralized, privacy-preserving and low-communication way. Following McMahan et al. 2017a, many existing FL algorithms aim at minimizing the following arithmetic average performance across the clients:

$$\min_{\boldsymbol{\theta}} f(\boldsymbol{\theta}) = \sum_{i=1}^{n} \lambda_i f_i(\boldsymbol{\theta}), \tag{1.1}$$

where the averaging weights $\{\lambda_i\}_{i=1}^n$ sum to 1. Different FL algorithms can be employed to solve the above problem, and the output will be one single model parameter $\boldsymbol{\theta}^*$ for all the participating clients. As mentioned earlier, the data distribution across clients can be heterogeneous, meaning that the data distribution can vary from one client to another client. Consequently, the single model parameter $\boldsymbol{\theta}^*$ may not result in a satisfying performance for all clients. Personalized FL algorithms have been proposed to address this challenge.

1.1.2 Personalized FL

Model personalization (Evgeniou and Pontil 2004; Li et al. 2020b) has been used as a technique to ensure a satisfying model performance for all participating clients. Instead of learning one single global model for all clients, personalized FL learns one model specifically for each client, which is adapted to its local data distribution. Ditto (Li et al. 2020b) is a personalized FL algorithm: Assuming $\boldsymbol{\theta}^*$ is the solution of the problem (1.1), Ditto solves the following problem for each client i to find its personalized model parameter:

$$\min_{\boldsymbol{\theta}_i} g_i(\boldsymbol{\theta}_i, \boldsymbol{\theta}^*) := f_i(\boldsymbol{\theta}_i) + \frac{\lambda}{2}\|\boldsymbol{\theta}_i - \boldsymbol{\theta}^*\|^2,$$

where λ is a regularization parameter tuning the level of personalization for clients. Mean-regularized multi-task learning (MR-MTL) (Evgeniou and Pontil 2004) is another method for personalized FL, which has less computational overhead for personalization compared to Ditto. The objective function of MR-MTL can be expressed as:

$$\min_{\boldsymbol{\theta}_i, i\in\{1,\dots,n\}} \sum_{i=1}^{n} g_i(\boldsymbol{\theta}_i) \quad \text{with} \quad g_i(\theta_i) = f_i(\boldsymbol{\theta}_i) + \frac{\lambda}{2}\|\boldsymbol{\theta}_i - \bar{\boldsymbol{\theta}}\|_2^2,$$

where $\bar{\boldsymbol{\theta}} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{\theta}_i$ is the average model parameter across clients. With $\lambda = 0$, MR-MTL reduces to local training. A larger regularization term λ encourages local models to be closer to each other, i.e. more federation. However, MR-MTL may not recover the solution of problem (1.1) as $\lambda \to \infty$. We refer to algorithm A1 in (Liu et al. 2022) for more details about MR-MTL.

1.1.3 Clustered FL

While personalized FL has been proposed to address mild data heterogeneity in FL, clustered FL has been proposed for scenarios where clients can be grouped into some clusters based on their data similarity, and there is a high data heterogeneity across different clusters. In this scenario, due to the high level of data heterogeneity in the system, learning one global model leads to a poor model performance for all clients. Proposed clustered FL algorithms aim at detecting the true underlying cluster structure of clients followed by learning one model for each cluster (Briggs et al. 2020; Ghosh et al. 2020; Mansour et al. 2020; Ruan and Joe-Wong 2021; Sattler et al. 2019; Werner et al. 2023). Some existing algorithms group clients based on their loss values (Dwork et al. 2012; Ghosh et al. 2020; Liu et al. 2022; Mansour et al. 2020; Ruan and Joe-Wong 2021), i.e. client i gets assigned to the cluster model resulting in the smallest train loss on its local dataset $\mathcal{D}_i$. Some other algorithms group them based on their model updates (based on e.g., their euclidean distance (Briggs et al. 2020; Werner et al. 2023) or cosine similarity (Sattler et al. 2019)). As shown by Werner et al. 2023, the algorithms are prone to clustering errors in the early rounds of FL training – due to gradient stochasticity, model initialization or the form of loss functions far from their optima – which can even propagate in the subsequent rounds.

1.1.4 Performance fairness in FL

Performance fairness across clients is desired to ensure a satisfying performance for the trained model (e.g., test accuracy) across all clients. Depending on the level of data heterogeneity in the system, performance fairness may be achievable by training one global model for all clients or by model personalization. If there is a slight data heterogeneity across clients, learning one fair global model for all is a possible task. For instance, Mohri et al. 2019 proposed Agnostic Federated Learning (AFL), which replaces the arithmetic average in problem (1.1) with the largest loss function of all clients in each round and finds the common model parameter $\boldsymbol{\theta}^*$ through a minimax optimization framework. Similarly, Li et al. 2020a proposed an algorithm called TERM based on weighted average of clients' loss functions. Also, q-Fair FL (q-FFL) (Li et al. 2020d) is an algorithm inspired by α-fairness in wireless networks. Similarly, the work by Zhang et al. 2023 proposed PropFair based on Proportional Fairness, and showed that all the aforementioned fair FL algorithms can be unified into a generalized mean framework.

1.2 Differential privacy (DP)

Differential privacy (Dwork et al. 2006a,b; Dwork and Roth 2014) is a standard method for ensuring data privacy in ML, and it has also been combined with FL in order to provide rigorous privacy guarantees to the participating clients. There are different definitions of DP, as explained in the following.

1.2.1 ϵ-DP

The first and main definition of DP is ϵ-DP, which is explained in the following definition:

Definition 1 (ϵ-DP, Dwork et al. 2006b)**.** *A randomized mechanism $\mathcal{M} : \mathcal{A} \rightarrow \mathcal{R}$ with domain $\mathcal{A}$ and range $\mathcal{R}$ satisfies ϵ-DP if for any two adjacent inputs $\mathcal{D}$, $\mathcal{D}' \in \mathcal{A}$, and for any subset of outputs $\mathcal{S} \subseteq \mathcal{R}$ it holds that:*

$$Pr[\mathcal{M}(\mathcal{D}) \in \mathcal{S}] \leq e^{\epsilon} Pr[\mathcal{M}(\mathcal{D}') \in \mathcal{S}].$$

The notion of adjacency of datasets is typically used to capture the contribution of a minor change in the input on the output. Most prior work on differentially private machine learning use *record-level* notion of adjacency (Abadi et al. 2016). In record-level adjacency, two datasets $\mathcal{D}$ and $\mathcal{D}'$ are called record-level adjacent if $\mathcal{D}$ can be constructed from $\mathcal{D}'$ by replacing a single data sample from $\mathcal{D}'$, i.e. they differ in only one data point and $|\mathcal{D}| = |\mathcal{D}'|$.

Laplace mechanism is a mechanism used for making ϵ-DP version of algorithms from their non-DP, deterministic versions. The mechanism randomizes the output of an arbitrary query f applied to a dataset $\mathcal{D}$ as:

$$\mathbf{L}_{\epsilon} f(\mathcal{D}) = f(\mathcal{D}) + \mathcal{L}(0, \frac{\Delta_1 f}{\epsilon}),$$

where $\Delta_1 f$ is the l_1-sensitivity of the query f:

$$\Delta_1 f = \max_{\mathcal{D},\mathcal{D}'} \parallel f(\mathcal{D}) - f(\mathcal{D}') \parallel_1$$

The randomized output of the Laplace mechanism $\mathbf{L}_{\epsilon} f(\mathcal{D})$ satisfies ϵ-DP. Different ϵ-DP mechanisms can be composed with each other. The basic composition theorem about standard differential privacy (ϵ-DP) says that if f and g are respectively ϵ_1-DP and ϵ_2-DP, then simultaneous release of f and g is a $(\epsilon_1 + \epsilon_2)$-DP mechanism.

1.2.2 Approximate $(\epsilon, \delta)-$DP

An approximation of the above-mentioned definition is the approximate $(\epsilon, \delta)-$DP, as follows:

Definition 2 ((ϵ, δ)-DP (Dwork et al. 2006a)). *A randomized mechanism $\mathcal{M} : \mathcal{A} \to \mathcal{R}$ with domain $\mathcal{A}$ and range $\mathcal{R}$ satisfies (ϵ, δ)-DP if for any two adjacent inputs $\mathcal{D}$, $\mathcal{D}' \in \mathcal{A}$, which differ only by a single record (by replacement), and for any measurable subset of outputs $\mathcal{S} \subseteq \mathcal{R}$ it holds that*

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

Gaussian mechanism, which randomizes the output of a non-private computation f on a dataset $\mathcal{D}$ as $\mathbf{G}_{\sigma} f(\mathcal{D}) = f(\mathcal{D}) + \mathcal{N}(\mathbf{0}, \sigma^2)$, provides (ϵ, δ)-DP for a continuum of pairs (ϵ, δ): it is (ϵ, δ)-DP for all $\epsilon \in (0, 1)$ and $\sigma > \frac{\sqrt{2\ln(1.25/\delta)}}{\epsilon} \Delta_2 f$, where $\Delta_2 f = \max_{\mathcal{D},\mathcal{D}'} \parallel f(\mathcal{D}) - f(\mathcal{D}') \parallel_2$ is the l_2-sensitivity of the query f with respect to its input. For a fixed δ, the guaranteed privacy parameter ϵ resulting from running Gaussian mechanism depends on the quantity $z = \frac{\sigma}{\Delta_2 f}$, which is called *noise scale*. Data subsampling can enhance the privacy guarantees of a $(\epsilon, \delta)-$DP mechanism. We have the following property about privacy amplification by subsampling.

Lemma 1 (Privacy Amplification by Subsampling (Dwork et al. 2006a)). *If $\mathcal{M} : \mathcal{X}^m \to \mathcal{Y}$ satisfies (ϵ, δ)-DP, with respect to the replacement relationship on sets of size m, $\mathcal{M}' = \mathcal{M} \circ \mathcal{S} : \mathcal{X}^n \to \mathcal{Y}$ satisfies $\big(log(1 + m(e^{\epsilon} - 1)/n),\ \ m\delta/n\big)$-DP $(n \geq m)$, where $\mathcal{S}$ is random sampling operator, which samples m out of n uniformly.*

A useful property of the relaxed (ϵ, δ)-DP is "advanced composition", which can be used for accounting the privacy guarantees of k-fold composition of (ϵ, δ)-DP mechanisms:

Lemma 2 (Advanced Composition Property (Dwork et al. 2010)). $\forall \epsilon, \delta, \delta' > 0, k \in \mathbb{N}$, *the family of (ϵ, δ)-DP mechanisms satisfies $(k\epsilon(e^{\epsilon} - 1) + \epsilon\sqrt{2k\ln(1/\delta')}, k\delta + \delta')$-DP under k-fold adaptive composition.*

Despite having the above property, we use the moments accountant (Abadi et al. 2016) for accounting the composition of some $(\epsilon, \delta)-$DP mechanisms. The accountant returns a tighter privacy guarantee than the advanced composition property above.

1.2.3 Rényi DP

There is another relaxed definition of DP, named Rényi DP (RDP), which we use for tight privacy accounting of composition of some heterogeneous private mechanisms. It is defined as follows:

Definition 3 (Rényi Differential Privacy (RDP) (Mironov 2017)). *A randomized mechanism $\mathcal{M} : \mathcal{A} \to \mathcal{R}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ satisfies (α, ϵ)-RDP with order α if for any two adjacent inputs $\mathcal{D}$, $\mathcal{D}' \in \mathcal{A}$, which differ only in a single record (by replacement), it satisfies*

$$D_{\alpha}\big(\mathcal{M}(\mathcal{D}) || \mathcal{M}(\mathcal{D}')\big) \leq \epsilon,$$

where $D_\alpha(P||Q)$ is the Rényi divergence between distributions P and Q:

$$D_\alpha(P||Q) := \frac{1}{\alpha-1}\log \mathbb{E}_{x\sim p}\left[\left(\frac{P(x)}{Q(x)}\right)^{\alpha-1}\right].$$

For $\alpha = 1$, we have $D_1(P||Q) := \mathbb{E}_{x\sim p}\left[\log\left(\frac{P(x)}{Q(x)}\right)\right]$, which is the KL divergence between P and Q. The following lemma is about conversion of (α, ϵ)-RDP to the approximate (ϵ, δ)-DP.

Lemma 3 (Converting RDP (Bun and Steinke 2016; Canonne et al. 2020)). *If a mechanism $\mathcal{M}$ satisifes $(\alpha, \epsilon(\alpha))$-RDP, then for any $\delta > 0$, it satisfies $(\epsilon(\delta), \delta)$-DP, where*

$$\epsilon(\delta) = \inf_{\alpha>1} \epsilon(\alpha) + \frac{1}{\alpha-1}\log\left(\frac{1}{\alpha\delta}\right) + \log\left(1-\frac{1}{\alpha}\right).$$

RDP is conveniently linearly composable, as explained in the following theorem.

Theorem 1 (Linear Composition of RDP (Mironov 2017)). *If mechanism $\mathcal{M}_i$ satisfies $(\alpha, \epsilon_i)-RDP$ for $i = 1, \ldots, k$, then the composed mechanism $\mathcal{M}_1 \circ \ldots \circ \mathcal{M}_k$ satisfies $(\alpha, \sum_{i=1}^k \epsilon_i)-RDP$.*

Therefore, if an algorithm has E steps (e.g., E batch gradient update steps) and each satisfies (α, ϵ)-RDP, the algorithm will satisfy $(\alpha, E\epsilon)$-RDP. It can also be used for composition of *heterogeneous* private mechanisms, as in the above theorem. There are different implementations of the RDP privacy accountant, e.g., the TensorFlow Privacy used in this thesis (McMahan et al. 2019).

1.2.4 Zero concentrated DP (zCDP)

Another relaxed definition of differential privacy is zero concentrated differential privacy (zCDP). Being ρ-zCDP is equivalent to being $(\alpha, \rho\alpha)$-RDP simultaneously for all $\alpha > 1$ (Bun and Steinke 2016). Therefore, standard RDP accountants, e.g., the aforementioned TensorFlow Privacy RDP accountant, can be used for accounting zCDP mechanisms as well.

1.2.5 Exponential mechanism for private selection

Exponential Mechanism is a standard method for private selection from a set of candidates. The selection is based on a score, which is assigned to every candidate (Rogers and Steinke 2021). Let us assume there is a private dataset $\mathcal{D}$ and a score function $s : \mathcal{D} \times [M] \to \mathbb{R}$, which evaluates a set of M candidate models on the dataset $\mathcal{D}$. The goal is to select the candidate with the highest score, i.e. $\text{argmax}_{m\in[M]}\, s(\mathcal{D}, m)$. Exponential mechanism performs this selection privately: It sets the probability of choosing any candidate $m \in [M]$ as:

$$\texttt{Pr}[m] = \frac{\exp(\frac{\epsilon_{\text{select}}}{2\Delta} \cdot s(\mathcal{D}, m))}{\sum_{m' \in [M]} \exp(\frac{\epsilon_{\text{select}}}{2\Delta} \cdot s(\mathcal{D}, m'))},$$

where Δ is the sensitivity of the scoring function s to the replacement of a data sample in $\mathcal{D}$. It can be shown that the private selection performed by exponential mechanism satisfies $\frac{1}{8}\epsilon^2_{\text{select}}$-zCDP (Bun and Steinke 2016). From the last paragraph, we know it also satisfies $(\alpha, \frac{\alpha}{8}\epsilon^2_{\text{select}})$-RDP for $\alpha > 1$. One can implement exponential mechanism by noisy selection with Gumbel noise: add independent noises from Gumbel distribution with scale $\frac{2\Delta}{\epsilon_{\text{select}}}$ to candidate scores $s(\mathcal{D}, m)$, for $m \in [M]$, and select the candiate with the maximum noisy score. The larger the sensitivity Δ to replacement of a single sample in $\mathcal{D}$, the larger the required noise scale.

1.2.6 Differentially private SGD (DPSGD)

DP was initially designed to limit the amount of information leakage about individual data when releasing an aggregated information from a database. Abadi et al. 2016 extended DP to DL by introducing DPSGD, which is widely used for private training of ML models. DPSGD applies an ℓ_2 norm clipping to every sample gradient in a batch of sample gradients and applies Gaussian mechanism to their average by adding a calibrated noise to it. The model parameters are iteratively updated with the privatized batch gradients, and the final DP guarantee is retained by sequentially composing the Gaussian mechanisms with DP guarantees (Abadi et al. 2016; Mironov et al. 2019). There is always a tradeoff between privacy and utility of DP trained models. Improving the tradeoff is constantly an important objective in related works.

1.3 Outline and contributions

Each of the following chapters are centered around one of my full-length articles. The main contributions of the following chapters are summarized as follows:

In Chapter 2, we study the optimization theory behind FL and

- We connect FL with operator splitting theory.
- We shed new insights on existing algorithms, refine their convergence analysis and suggest new algorithmic variants.
- Present a unification of current algorithms and propose strategies for their acceleration.

In Chapter 3, we augment FL with DP for providing rigorous data privacy guarantees to clients.

- We show the effect of privacy parameter ϵ and batch/dataset size on the noise level in clients' updates, which are uploaded to the server.
- We propose "Robust-HDP", a robust algorithm with a noise-aware aggregation strategy for heterogeneous DPFL, where each client has its own privacy preference.
- 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.
- In various privacy heterogeneity scenarios across clients, we show that Robust-HDP improves utility and convergence speed while respecting clients' privacy.

Finally, in Chapter 4, we study DPFL with structured data heterogeneity across clients.

- We propose a DP clustered FL algorithm (R-DPCFL), which combines information from both clients' model updates and their loss values. The algorithm is robust and achieves high-quality clustering of clients, even in the presence of DP noise in the system.
- We theoretically prove that increasing clients' batch sizes in the first round (and decreasing them in the subsequent rounds) reduces the amount of noise in clients' model updates at the end of the first round and enhances the server's ability to cluster the clients based on them.
- We show that utilizing sufficiently large client batch sizes in the first round (and sufficiently small batch sizes in the next rounds) enables super-linear convergence rate for learning a GMM on clients' model updates. This enables us to soft cluster clients using a GMM with a low computational overhead.
- We extensively evaluate our proposed algorithm for DP clustered FL across diverse datasets and scenarios, and demonstrate its effectiveness in detecting the underlying cluster structure of clients, which leads to an overall utility improvement for the system.

Chapter 2

An Operator Splitting View of Federated Learning

2.1 Preface

Over the past few years, the federated learning (FL) community has witnessed a proliferation of new FL algorithms. However, our understating of the theory of FL is still fragmented, and a thorough, formal comparison of these algorithms remains elusive. Motivated by this gap, we show that many of the existing FL algorithms can be understood from an operator splitting point of view. This unification allows us to compare different algorithms with ease, to refine previous convergence results and to uncover new algorithmic variants. In particular, our analysis reveals the vital role played by the step size in FL algorithms. The unification also leads to a streamlined and economic way to accelerate FL algorithms, without incurring any communication overhead. We perform numerical experiments on both convex and nonconvex models to validate our findings.

2.2 Introduction

The accompanying (moderate) computational power of modern smart devices such as phones, watches, home appliances, cars, etc., and the enormous data accumulated from their interconnected ecosystem have fostered new opportunities and challenges to train/tailor modern deep models. Federated learning (FL), as a result, has recently emerged as a massively distributed framework that enables training a shared or personalized model without infringing clients' privacy. Tremendous progress has been made since the seminal work of McMahan et al. (2017b), including algorithmic innovations (Huo et al. 2020; Li et al. 2020c; Pathak and Wainwright 2020; Reddi et al. 2020; Wang et al. 2020; Yurochkin et al. 2019), convergence analysis (Gorbunov et al. 2021; Khaled et al. 2020a,b; Li et al. 2020e; Malinovskiy et al. 2020), personalization (Deng et al. 2020; Diao et al. 2021; Dinh et al. 2020; Mansour et al. 2020; Zhang et al. 2021), privacy protection (Augenstein et al. 2020; Nasr et al. 2019), model robustness (Bagdasaryan et al. 2020; Bhagoji et al. 2019;

Reisizadeh et al. 2020; Sun et al. 2019), fairness (Hu et al. 2022; Malekmohammadi and Yu 2025; Mohri et al. 2019; Zhang et al. 2023), standardization (Caldas et al. 2019; He et al. 2020), applications (Smith et al. 2017), just to name a few. We refer to the excellent surveys of Kairouz et al. 2021; Li et al. 2019; Yang et al. 2019 and the references therein for the current state of affairs in FL.

The main goal of this chapter is to take a closer examination of some of the most popular FL algorithms, including FedAvg (McMahan et al. 2017b), FedProx (Li et al. 2020c), and FedSplit (Pathak and Wainwright 2020), by connecting them with the well-established theory of operator splitting in optimization. In particular, we show that FedAvg corresponds to forward-backward splitting and we demonstrate a trade-off in its step size and number of local epochs, while a similar phenomenon has also been observed in (Charles and Konečný 2021; Li et al. 2020e; Malinovskiy et al. 2020; Pathak and Wainwright 2020). FedProx, on the other hand, belongs to backward-backward splitting, or equivalently, forward-backward splitting on a related but regularized problem, which has been somewhat overlooked in the literature. Interestingly, our results reveal that the recent personalized model in (Dinh et al. 2020) is exactly the problem that FedProx aims to solve. Moreover, we show that when the step size in FedProx diminishes (sublinearly fast), then it actually solves the original problem, which is contrary to the observation in (Pathak and Wainwright 2020) where the step size is fixed and confirms again the importance of step size in FL. For FedSplit, corresponding to Peaceman-Rachford splitting (Lions and Mercier 1979; Peaceman and Rachford 1955), we show that its convergence (in theory) heavily hinges on the strong convexity of the objective and hence might be less stable for nonconvex problems.

Inspecting FL through the lens of operator splitting theory also allows us to immediately uncover new FL algorithms, by adapting existing splitting algorithms. Indeed, we show that Douglas-Rachford splitting (Douglas and Rachford 1956; Lions and Mercier 1979) (more precisely the method of partial inverse of Spingarn 1985) yields a (slightly) slower but more stable variant of FedSplit, while at the same time shares a close connection to FedProx: the latter basically freezes the update of a dual variable in the former. We also propose FedRP by combining the reflector in FedSplit and the projection (averaing) in FedAvg and FedProx, and extend an algorithm of Bauschke and Kruk (2004). We improve the convergence analysis of FedRP and empirically demonstrate its competitiveness with other FL algorithms. We believe these results are just the tip of the ice-berg and much more progress can now be expected: the marriage between FL and operator splitting theory simply opens a range of new possibilities to be explored.

Our holistic view of FL algorithms suggests a very natural unification, building on which we show that the aforementioned algorithms reduce to different parameter settings in a grand scheme. We believe this is an important step towards standardizing FL from an algorithmic point of view, in addition to standard datasets, models and evaluation protocols as already articulated in (Caldas et al. 2019; He et al. 2020). Our unification also allows one to compare and implement different FL algorithms with ease, and more importantly to accelerate them in a streamlined and economic fashion. Indeed, for the first time we show that Anderson acceleration (Anderson 1965; Fu et al. 2020), originally proposed for nonlinear fixed-point iteration, can be adapted to accelerate existing FL algorithms and we provide practical implementations that do not incur any communication overhead. We perform experiments on both convex and nonconvex models to validate our findings,

and compare the above-mentioned FL algorithms on an equal footing.

We proceed in §2.3 with some background introduction. Our main contributions are presented in §2.4, where we connect FL with operator splitting, shed new insights on existing algorithms, suggest new algorithmic variants and refine convergence analysis, and in §2.5, where we present a unification of current algorithms and propose strategies for further acceleration. We conclude in §2.6 with some future directions. All proofs are deferred to Section C in the appendix.

2.3 Background

In this section, we recall the federated learning (FL) framework introduced in Chapter 1. We consider n clients (edge devices), where the i-th client aims at minimizing a function $f_i : \mathbb{R}^p \to \mathbb{R}, i = 1, \dots, n$, defined on a shared model parameter $\boldsymbol{\theta} \in \mathbb{R}^p$. Typically, each client function f_i depends on the respective client's local (private) dataset $\mathcal{D}_i$. The main goal in FL is to *collectively and efficiently* optimize *individual* objectives $\{f_i\}$ in a decentralized, privacy-preserving and low-communication way. Following McMahan et al. (2017b), many existing FL algorithms fall into the natural formulation that optimizes the (arithmetic) average performance:

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^p} \; f(\boldsymbol{\theta}), \qquad \text{where} \qquad f(\boldsymbol{\theta}) := \sum_{i=1}^{n} \lambda_i f_i(\boldsymbol{\theta}). \tag{2.1}$$

The weights λ_i here are nonnegative and w.l.o.g. sum to 1. Below, we assume they are specified *beforehand* and fixed throughout.

To facilitate our discussion, we recall the Moreau envelope and proximal map of a function[1] $f : \mathbb{R}^p \to \mathbb{R} \cup \{\infty\}$, defined respectively as:

$$\mathsf{M}_f^{\eta}(\boldsymbol{\theta}) = \min_{\mathbf{x}} \; f(\mathbf{x}) + \tfrac{1}{2\eta}\|\mathbf{x} - \boldsymbol{\theta}\|_2^2, \qquad \mathsf{P}_f^{\eta}(\boldsymbol{\theta}) = \operatorname*{argmin}_{\mathbf{x}} \; f(\mathbf{x}) + \tfrac{1}{2\eta}\|\mathbf{x} - \boldsymbol{\theta}\|_2^2,$$

where $\eta > 0$ is a parameter acting similarly as the step size. We also define the reflector

$$\mathsf{R}_f^{\eta}(\boldsymbol{\theta}) = 2\mathsf{P}_f^{\eta}(\boldsymbol{\theta}) - \boldsymbol{\theta}.$$

As shown in Figure 2.1, when $f = \iota_C$ is the indicator function of a (closed) set $C \subseteq \mathbb{R}^p$, $\mathsf{P}_f^{\eta}(\boldsymbol{\theta}) \equiv \mathsf{P}_C(\boldsymbol{\theta})$ is the usual (Euclidean) projection onto the set C while $\mathsf{R}_f^{\eta}(\boldsymbol{\theta}) \equiv \mathsf{R}_C(\boldsymbol{\theta})$ is the reflection of $\boldsymbol{\theta}$ w.r.t. C.

We are now ready to reformulate our main problem of interest (2.1) in a product space:

$$\min_{\boldsymbol{\Theta} \in H} \; \mathsf{f}(\boldsymbol{\Theta}) = \langle \mathbf{1}, \mathbf{f}(\boldsymbol{\Theta}) \rangle, \qquad \text{where} \qquad \mathbf{f}(\boldsymbol{\Theta}) := \big(f_1(\boldsymbol{\theta}_1), \dots, f_n(\boldsymbol{\theta}_n)\big), \tag{2.2}$$

$\mathbf{1}$ is the vector of all 1s,

$$H := \{\boldsymbol{\Theta} = (\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_n) \in \mathbb{R}^{pn} : \; \boldsymbol{\theta}_1 = \cdots = \boldsymbol{\theta}_n\}$$

[1] We allow the function f to take ∞ when the input is out of our domain of interest.

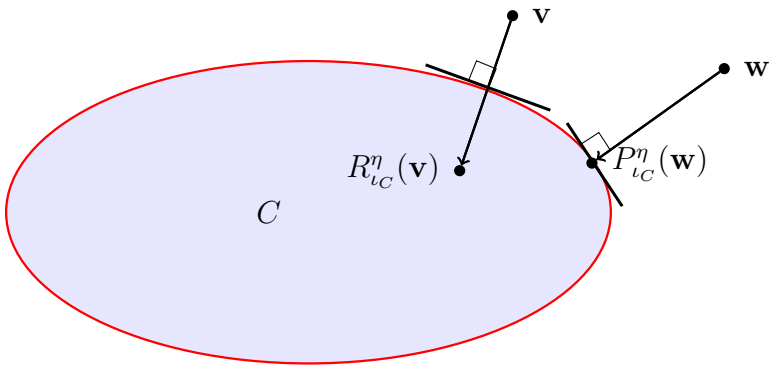


Figure 2.1: When $f = \iota_C$ is the indicator function of a (closed) set C, $\mathsf{P}^{\eta}_f(\mathbf{w})$ is the Euclidean projection of $\mathbf{w}$ onto the set C. Similarly, $\mathsf{R}^{\eta}_f(\mathbf{v})$ is the Euclidean reflection of $\mathbf{v}$ w.r.t. the set C.

and we equip with the inner product

$$\langle \boldsymbol{\Theta}, \mathbf{z} \rangle = \langle \boldsymbol{\Theta}, \mathbf{z} \rangle_{\boldsymbol{\lambda}} := \sum_i \lambda_i \boldsymbol{\theta}_i^\top \mathbf{z}_i.$$

Note that the complement $H^\perp = \{\boldsymbol{\Theta} : \sum_i \lambda_i \boldsymbol{\theta}_i = \mathbf{0}\}$, and

$$\mathsf{P}_H(\boldsymbol{\Theta}) = (\bar{\boldsymbol{\theta}}, \ldots, \bar{\boldsymbol{\theta}}), \quad \text{where} \quad \bar{\boldsymbol{\theta}} := \sum\nolimits_i \lambda_i \boldsymbol{\theta}_i, \text{ and } \mathsf{R}_H(\boldsymbol{\Theta}) = (2\bar{\boldsymbol{\theta}} - \boldsymbol{\theta}_1, \ldots, 2\bar{\boldsymbol{\theta}} - \boldsymbol{\theta}_n).$$

We define the (forward) gradient update map w.r.t. a (sub)differentiable function f:

$$\mathsf{G}^{\eta}_f := \mathrm{Id} - \eta \cdot \partial f, \qquad \boldsymbol{\theta} \mapsto \boldsymbol{\theta} - \eta \cdot \partial f(\boldsymbol{\theta}).$$

When f is differentiable and convex, we note that $\mathsf{R}^{\eta}_f = \mathsf{G}^{\eta}_f \circ \mathsf{P}^{\eta}_f$ and $\nabla \mathsf{M}^{\eta}_f = \frac{\mathrm{Id} - \mathsf{P}^{\eta}_f}{\eta}$.

2.4 Federated learning as operator splitting

Following Pathak and Wainwright (2020), in this section we interpret existing FL algorithms (FedAvg , FedProx , FedSplit) from the operator splitting point of view. We reveal the importance of the step size, obtain new convergence guarantees, and uncover some new and accelerated algorithmic variants.

2.4.1 FedAvg as forward-backward splitting

The FedAvg algorithm of McMahan et al. (2017b) is essentially a k-step version of the forward-backward splitting approach of Bruck (1977):

$$\boldsymbol{\Theta}_{t+1} \leftarrow \mathsf{P}_H \mathsf{G}^{\eta_t}_{\mathsf{f},k} \boldsymbol{\Theta}_t, \qquad \text{where} \qquad \mathsf{G}^{\eta_t}_{\mathsf{f},k} := \underbrace{\mathsf{G}^{\eta_t}_{\mathsf{f}} \circ \mathsf{G}^{\eta_t}_{\mathsf{f}} \circ \cdots \circ \mathsf{G}^{\eta_t}_{\mathsf{f}}}_{k \text{ times}}, \tag{2.3}$$

where we perform k steps of (forward) gradient updates w.r.t. f followed by 1 (backward) proximal update w.r.t. H. To improve efficiency, McMahan et al. (2017b) replace $\mathsf{G}_{\mathsf{f}}^{\eta_t}$ with $\hat{\mathsf{G}}_{\mathsf{f}}^{\eta_t}$ where the (sub)gradient is approximated on a minibatch of training data and P_H with $\hat{\mathsf{P}}_H$ where we only average over a chosen subset $I_t \subseteq [n] := \{1, \dots, n\}$ of clients. Note that the forward step $\mathsf{G}_{\mathsf{f},k}^{\eta_t}$ is performed in parallel at the client side while the backward step P_H is performed at the server side.

The number of local steps k turns out to be a key factor: Setting $k = 1$ reduces to the usual (stochastic) forward-backward algorithm and enjoys the well-known convergence guarantees. On the other hand, setting $k = \infty$ (and assuming convergence on each local function f_i) amounts to (repeatedly) averaging the chosen minimizers of f_i's, and eventually converges to the average of minimizers of all n client functions. In general, the performance of the fixed point solution of FedAvg appears to have a dependency on the number of local steps k and the step size η. Let us illustrate this with quadratic functions $f_i(\boldsymbol{\theta}) = \frac{1}{2}\|A_i\boldsymbol{\theta} - \mathbf{b}_i\|_2^2$ and non-adaptive step size $\eta_t \equiv \eta$, where we obtain the fixed-point $\boldsymbol{\theta}^*_{\mathsf{FedAvg}}(k)$ of FedAvg in closed-form (Pathak and Wainwright 2020):

$$\boldsymbol{\theta}^*_{\mathsf{FedAvg}}(k) = \big(\textstyle\sum_{i=1}^n \frac{1}{k}\sum_{j=0}^{k-1} A_i^\top A_i (I - \eta A_i^\top A_i)^j\big)^{-1}\big(\sum_{i=1}^n \frac{1}{k}\sum_{j=0}^{k-1}(I - \eta A_i^\top A_i)^j A_i^\top \mathbf{b}_i\big).$$

For small η, we may apply Taylor expansion and ignore the higher order terms:

$$\frac{1}{k}\sum_{j=0}^{k-1}(I - \eta A_i^\top A_i)^j \approx \frac{1}{k}\sum_{j=0}^{k-1}(I - j\eta A_i^\top A_i) = I - \frac{\eta(k-1)}{2}A_i^\top A_i. \tag{2.4}$$

Thus, we observe that the fixed point $\boldsymbol{\theta}^*_{\mathsf{FedAvg}}(k)$ of FedAvg depends on $\eta(k-1)$, up to higher order terms. In particular, when $k = 1$, the fixed point does not depend on η (as long as it is small enough to guarantee convergence), but for $k > 1$, the solution that FedAvg converges to does depend on η. Moreover, the final performance is almost completely determined by $\eta(k-1)$ in this quadratic setting when η is small.

Figure 2.2 illustrates a trade-off where larger local epochs k and learning rate η leads to faster convergence (in terms of communication rounds) during the early stages, at the cost of a worse final solution. Similar results are also reported by e.g., Charles and Konečný 2021; Li et al. 2020e; Malinovskiy et al. 2020; Pathak and Wainwright 2020. As observed, when η is fixed, larger local epochs k leads to faster convergence (in terms of communication rounds) during the early stages, at the cost of a worse final solution. When local epochs k is fixed, same trade-off holds true for increasing η.

In Figure 2.3, contrary to what we observe in the previous convex experiments where fixed learning rate with less local epochs leads to a better final solution at the expense of slower convergence, we do not see similar trade-offs in this nonconvex setting, (at least) given the computation we can afford. This is probably because the solution we get is still in early stage due to the combined complexity of CNN and CIFAR-10 dataset, compared to the relatively simple least squares and logistic regression we had above.

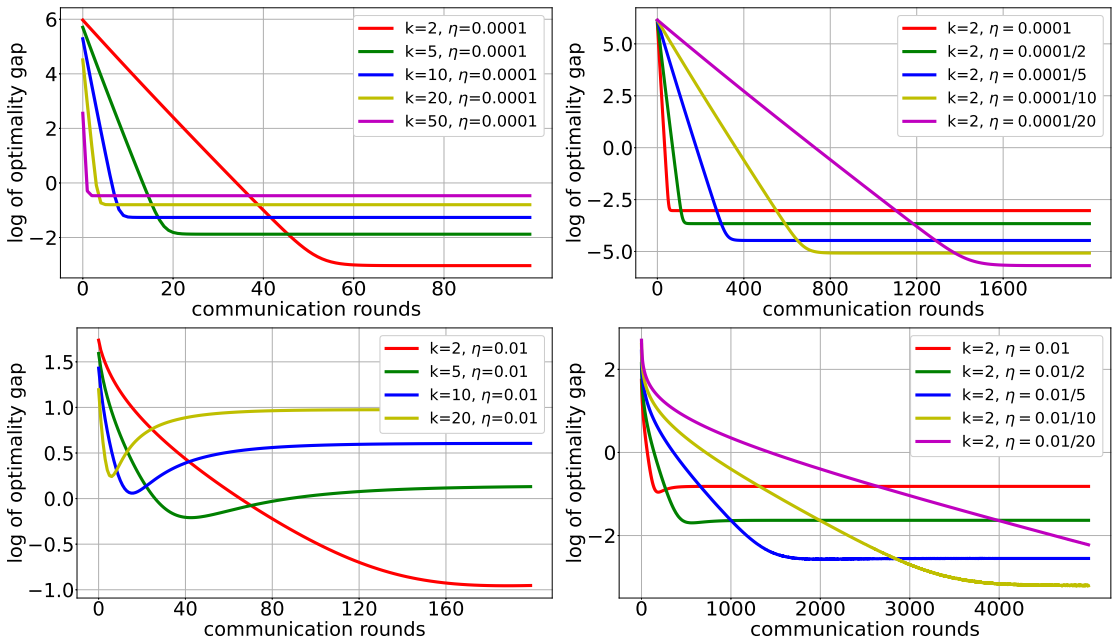


Figure 2.2: Convergence of FedAvg when changing k or η. Top: least squares; Bottom: logistic regression.

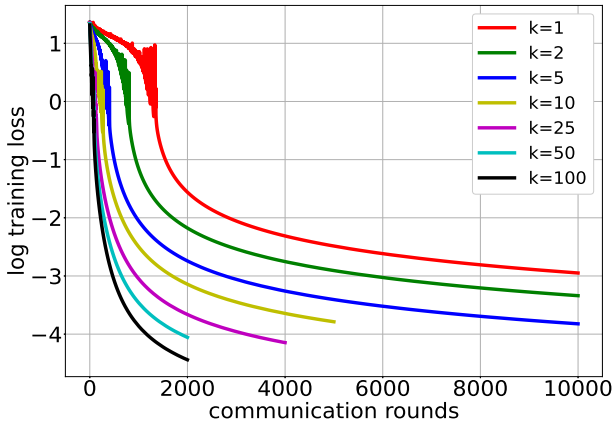


Figure 2.3: Running FedAvg on CIFAR-10 with different local epochs k. The number of clients is 10, local learning rate $\eta = 0.1$. Data points are sampled and distributed to clients non-uniformly.

In Figure 2.4, we run FedAvg in Equation (2.3) on least squares, logistic regression and a nonconvex CNN on the MNIST dataset (LeCun et al. 2010). The experiments are run with pre-determined numbers of communication rounds and different configurations of local epochs k and learning rates η (Complete details of our experimental setup are given in Section A of the appendix[2]). We examine the dependencies of FedAvg's final performance regarding k and

[2]All nonconvex experimental results in this chapter are averaged over 4 runs with different random seeds.

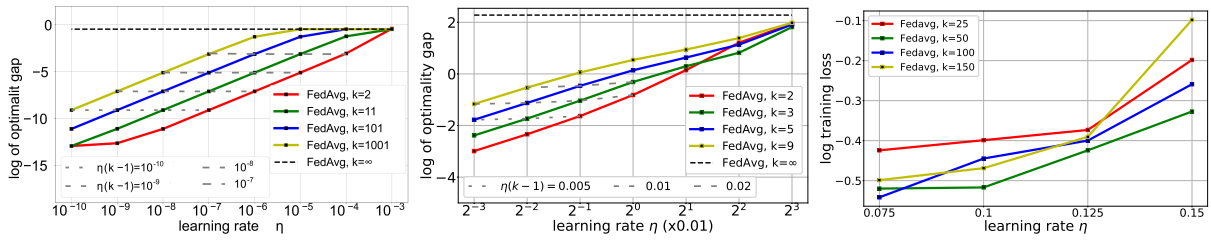


Figure 2.4: Optimality gap $\{f(\boldsymbol{\theta}^*_{\mathsf{FedAvg}}) - f^*\}$ or training loss $\{f(\boldsymbol{\theta}^*_{\mathsf{FedAvg}})\}$ of (approximate) fixed-point solutions of FedAvg for different learning rates η and local epochs k. Different colored lines are for different numbers of local epochs, and dashed lines for different product values $\eta(k-1)$. Left: least squares (closed-form solution); Middle: logistic regression (6000 communication rounds); Right: nonconvex CNN on the MNIST dataset (200 communication rounds).

η. We can conclude that, in general, smaller local epochs and learning rates gives better final solutions (assuming sufficient communication rounds to ensure convergence). Moreover, for least squares and logistic regression, the product $\eta(k-1)$ we derived in Equation (2.4) almost completely determines the final performance, when k and η are within a proper range. For the nonconvex CNN (Figure 2.4, right), especially with limited communications, the approximation in Equation (2.4) is too crude to be indicative.

2.4.2 FedProx as backward-backward splitting

The recent FedProx algorithm (Li et al. 2020c) replaces the gradient update in FedAvg with a proximal update:

$$\boldsymbol{\Theta}_{t+1} = \mathsf{P}_H \mathsf{P}_{\mathsf{f}}^{\eta_t} \boldsymbol{\Theta}_t, \tag{2.5}$$

where as before we may use a minibatch to approximate $\mathsf{P}_{\mathsf{f}}^{\eta}$ or select a subset of clients to approximate P_H. Written in this way, it is clear that FedProx is an instantiation of the backward-backward splitting algorithm of Lions (1978) and Passty (1979). In fact, this algorithm traces back to the early works of e.g. Auslender (1969), Cimmino (1938), and Lions and Temam (1966), sometimes under the name of the Barycenter method. It was also rediscovered by Yu (2013) and Yu et al. (2015) in the ML community under a somewhat different motivation. Pathak and Wainwright (2020) pointed out that FedProx does not solve the original problem (2.1). While technically correct, their conclusion did not take many other subtleties into account, which we explain next.

Following Bauschke et al. (2005), we first note that, with a constant step size $\eta_t \equiv \eta$, FedProx is actually equivalent as FedAvg but applied to a "regularized" problem:

$$\min_{\boldsymbol{\Theta} \in H} \ \tilde{\mathsf{f}}(\boldsymbol{\Theta}), \qquad \text{where} \qquad \tilde{\mathsf{f}}(\boldsymbol{\Theta}) := \langle \mathbf{1}, \mathsf{M}^{\eta}_{\mathsf{f}}(\boldsymbol{\Theta}) \rangle \,, \ \text{and} \ \ \mathsf{M}^{\eta}_{\mathsf{f}}(\boldsymbol{\Theta}) = \big(\mathsf{M}^{\eta}_{f_1}(\boldsymbol{\theta}_1), \ldots, \mathsf{M}^{\eta}_{f_n}(\boldsymbol{\theta}_n)\big). \tag{2.6}$$

Interestingly, Dinh et al. (2020) proposed exactly Equation (2.6) for the purpose of personalization, which we now realize is automatically achieved if we apply FedProx to the original formulation (2.1). Indeed, $\nabla \mathsf{M}_{\mathbf{f}}^{\eta}(\boldsymbol{\Theta}) = [\boldsymbol{\Theta} - \mathsf{P}_{\mathbf{f}}^{\eta}(\boldsymbol{\Theta})]/\eta$ hence $\mathsf{G}_{\tilde{f}}^{\eta}(\boldsymbol{\Theta}) = \boldsymbol{\Theta} - \eta \nabla \tilde{\mathsf{f}}(\boldsymbol{\Theta}) = \mathsf{P}_{\mathbf{f}}^{\eta}(\boldsymbol{\Theta}) = \big(\mathsf{P}_{f_1}^{\eta}(\boldsymbol{\theta}_1), \ldots, \mathsf{P}_{f_n}^{\eta}(\boldsymbol{\theta}_n)\big)$. This observation turns out to be crucial in understanding FedProx .

Indeed, a significant challenge in FL is client heterogeneity (a.k.a. non-iid distribution of data), where the individual client functions f_i may be very different (due to distinct client-specific data). But, we note that[3] as $\eta \to 0$, $\mathsf{M}_f^{\eta} \to f$ (pointwise or uniformly if f is Lipschitz) while $\mathsf{M}_f^{\eta} \to \min f$ as $\eta \to \infty$. In other words, a larger η in (2.6) "smoothens" heterogeneity in the sense that the functions $\mathsf{M}_{f_i}^{\eta}$ tend to have similar minimizers (while the minimum values may still differ). We are thus motivated to understand the convergence behaviour of FedProx, for small and large η, corresponding to the original problem (2.1) and the smoothened problem, respectively. In fact, we can even adjust η dynamically. Below, for simplicity, we assume full gradient (i.e. large batch size) although extensions to stochastic gradient are possible.

Theorem 2. *Assuming each client participates indefinitely, the step size η_t is bounded from below (i.e. $\liminf_t \eta_t > 0$), the client functions $\{f_i\}$ are convex, and homogeneous in the sense that they have a common minimizer, i.e. $\mathsf{F} := \bigcap_i \operatorname{argmin}_{\boldsymbol{\theta}_i} f_i(\boldsymbol{\theta}_i) \neq \emptyset$, then the iterates of* FedProx *converge to a point in F, which is a solution of* (2.1).

The homogeneity assumption might seem strong since it challenges the necessity of FL in the first place. However, we point out that (a) as a simplified limiting case it does provide insight on when client functions have close minimizers (i.e. homogeneous); (b) by using a large step size η FedProx effectively homogenizes the clients and its convergence thus follows[4]; (c) modern deep architectures are typically over-parameterized so that achieving null training loss on each client is not uncommon. However, FL is still relevant even in this case since it selects a model that works for *every* client and hence provides some regularizing effect; (d) even when the homogeneity assumption fails, $\mathsf{P}_{\mathsf{f}}^{\eta_t}$ still converges to a point that is in some sense closest to H (Bauschke et al. 2005).

The next result removes the homogeneity assumption by simply letting step size η_t diminish:

Theorem 3 ((Lions 1978; Passty 1979))**.** *Assuming each client participates in every round, the step size η_t satisfies $\sum_t \eta_t = \infty$ and $\sum_t \eta_t^2 < \infty$, and the functions $\{f_i\}$ are convex, then the* averaged *iterates $\bar{\boldsymbol{\Theta}}_t := \frac{\sum_{s=1}^{t} \eta_s \boldsymbol{\Theta}_s}{\sum_{s=1}^{t} \eta_s}$ of* FedProx *converge to a solution of the* original *problem* (2.1).

In Section C in the appendix, we give examples to show the tightness of the step size condition in Theorem 3. We emphasize that $\bar{\boldsymbol{\Theta}}_t$ converges to a solution of the original problem (2.1), not the regularized problem (2.6). The subtlety is that we must let the step size η_t approach 0 reasonably slowly, a possibility that was not discussed in Pathak and Wainwright (2020) where they always fixed the step size η_t to a constant η. Theorem 3 also makes intuitive sense, as $\eta_t \to 0$ slowly,

[3]These results are classic and well-known, see e.g. Rockafellar and Wets (1998).

[4]The consequence of this must of course be further investigated; see Dinh et al. (2020) and Li et al. (2020c).

we are effectively tracking the solution of the regularized problem (2.6) which itself tends to the original problem (2.1): recall that $\mathsf{M}_f^\eta \to f$ as $\eta \to 0$.

Even the ergodic averaging step in Theorem 3 can be omitted in some cases:

Theorem 4 ((Passty 1979)). *Under the same assumptions as in Theorem 3, if* f *is strongly convex or the solution set of* (2.1) *has nonempty interior, then the vanilla iterates* $\boldsymbol{\Theta}_t$ *of* FedProx *also converge.*

We remark that convergence is in fact linear for the second case. Nevertheless, the above two conditions are perhaps not easy to satisfy or verify in most applications. Thus, in practice, we recommend the ergodic averaging in Theorem 3 since it does not create additional overhead (if implemented incrementally) and in our experiments it does not slow down the algorithm noticeably.

There is in fact a quantitative relationship between the original problem (2.2) and the regularized problem (2.6), even in the absence of convexity:

Theorem 5 ((Yu et al. 2015)). *Suppose each* f_i *is* M_i*-Lipschitz continuous (and possibly nonconvex), then*

$$\forall \boldsymbol{\Theta}, \quad \mathsf{f}(\boldsymbol{\Theta}) - \tilde{\mathsf{f}}(\boldsymbol{\Theta}) \leq \textstyle\sum_i \lambda_i \frac{\eta M_i^2}{2}.$$

Thus, for a small step size η, FedProx, aiming to minimize the regularized function $\tilde{\mathsf{f}}$ is not quantitatively different from FedAvg that aims at minimizing f. This again reveals the fundamental importance of step size in FL. We remark that, following the ideas in Yu et al. (2015), we may remove the need of ergodic averaging for FedProx if the functions f_i are "definable" (which most, if not all, functions used in practice are). We omit the technical complication here since the potential gain does not appear to be practically significant.

In Figure 2.5, we show the effect of step size η on the convergence of FedProx, and compare the results with that of FedAvg on both convex and nonconvex models. We run FedProx with both fixed and diminishing step sizes. In the experiments with diminishing step sizes, we have set the initial value of η (i.e. η_0) to larger values - compared to the constant η values - to ensure that η does not get very small after the first few rounds. From the convex experiments (Figure 2.5, left and middle), one can see that FedProx with a fixed learning rate converges fast (in a few hundred rounds) to a suboptimal solution. In contrast, FedProx with diminishing η converges slower, but to better quality solutions. It is interesting to note that only when the conditions of Theorem 3 are satisfied (i.e. η diminishes neither too fast nor too slow), FedProx converges to a solution of the original problem (2.1), e.g. see the results for $\eta_t \propto 1/t$ which satisfies both conditions in Theorem 3. Surprisingly, for the nonconvex setting (Figure 2.5, right), the best results are achieved with larger learning rates. A similar observation about FedAvg in nonconvex settings was reported in (McMahan et al. 2017b, Figure 5 & 6) [5]. Moreover, from the results on both convex and nonconvex models, one can see that ergodic averaging does not affect the convergence rate of FedProx noticeably.

[5] Note that the results of FedAvg and FedProx for $\eta = 100$ and $100/\log(t)$ overlap with each other.

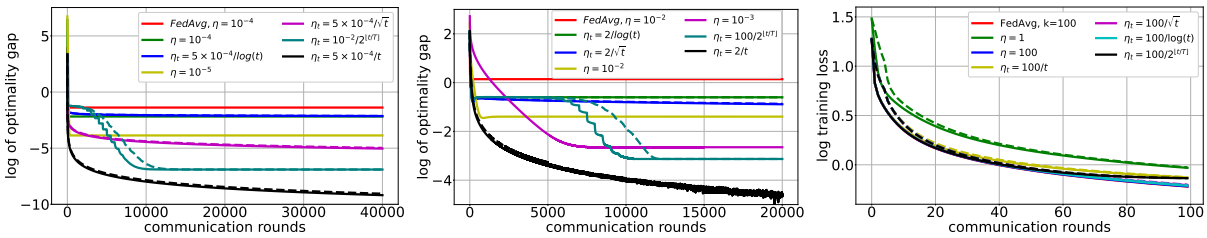


Figure 2.5: Effect of step size η and ergodic averaging on FedProx . Left: least squares; Middle: logistic regression; Right: CNN on MNIST. The dashed and solid lines with the same color show the results obtained with and without the ergodic averaging step in Theorem 3, respectively. For exponentially decaying η_t, we use period T equal to 500 for both least squares and logistic regression experiments, and 10 for CNN experiment.

2.4.3 FedSplit as Peaceman-Rachford splitting

Pathak and Wainwright (2020) introduced the FedSplit algorithm recently:

$$\boldsymbol{\Theta}_{t+1} = \mathsf{R}_H \mathsf{R}_{\mathsf{f}}^{\eta_t} \boldsymbol{\Theta}_t, \tag{2.7}$$

which is essentially an instantiation of the Peaceman-Rachford splitting algorithm (Lions and Mercier 1979; Peaceman and Rachford 1955). As shown by Lions and Mercier (1979), FedSplit converges *to a solution of* (2.1) if $\mathbf{f}$ is strictly convex, and the convergence rate is linear if $\mathbf{f}$ is strongly convex and smooth (and η is small). Pathak and Wainwright (2020) also studied the convergence behaviour of FedSplit when the reflector $\mathsf{R}_{\mathsf{f}}^{\eta}$ is computed approximately. However, we note that convergence behaviour of FedSplit is not known or widely studied for nonconvex problems. In particular, we have the following surprising result:

Theorem 6. *If the reflector R_f^{η} is a (strict) contraction, then f must be strongly convex.*

The converse is true if f is also smooth and η is small (Gabay 1983; Lions and Mercier 1979). Therefore, for non-strongly convex or nonconvex problems, we cannot expect FedSplit to converge linearly (if it converges at all).

2.4.4 FedPi as Douglas-Rachford splitting

A popular alternative to the Peaceman-Rachford splitting is the Douglas-Rachford splitting (Douglas and Rachford 1956; Lions and Mercier 1979), which, to our best knowledge, has not been applied to the FL setting. The resulting update, which we call FedPi, can be written succinctly as:

$$\boldsymbol{\Theta}_{t+1} = \frac{\boldsymbol{\Theta}_t + \mathsf{R}_H \mathsf{R}_{\mathsf{f}}^{\eta_t} \boldsymbol{\Theta}_t}{2}, \tag{2.8}$$

i.e. we simply average the current iterate and that of FedSplit evenly. Strictly speaking, the above algorithm is a special case of the Douglas-Rachford splitting and was rediscovered by Spingarn

(1983) under the name of partial inverse (hence our name FedPi). The moderate averaging step in (2.8) makes FedPi much more stable:

Theorem 7 ((Lions and Mercier 1979; Spingarn 1983))**.** *Assuming each client participates in every round, the step size $\eta_t \equiv \eta$ is constant, and the functions $\{f_i\}$ are convex, then the vanilla iterates $\boldsymbol{\Theta}_t$ of* FedPi *converge to a solution of the* original *problem* (2.1).

Compared to FedSplit, FedPi imposes less stringent condition on f_i. However, when f_i is indeed strongly convex and smooth, as already noted by Lions and Mercier (1979), FedPi will be slower than FedSplit by a factor close to $\sqrt{2}$ (assuming the constant step size is set appropriately for both). More importantly, it may be easier to analyze FedPi on nonconvex functions, as recently demonstrated in (Rockafellar 2019; Themelis and Patrinos 2020).

Interestingly, FedPi also has close ties to FedProx. Indeed, this is best seen by expanding the concise formula in (2.8) and introducing a "dual variable" $\mathbf{u}$ on the server side[6]:

$$\mathbf{z}_{t+1} \leftarrow \mathsf{P}_{\mathsf{f}}^{\eta}(\boldsymbol{\Theta}_t + \mathbf{u}_t) \tag{2.9}$$

$$\boldsymbol{\Theta}_{t+1} \leftarrow \mathsf{P}_H(\mathbf{z}_{t+1} - \mathbf{u}_t) \tag{2.10}$$

$$\mathbf{u}_{t+1} \leftarrow \mathbf{u}_t + \boldsymbol{\Theta}_{t+1} - \mathbf{z}_{t+1}. \tag{2.11}$$

From the last two updates (2.10) and (2.11) it is clear that $\mathbf{u}_{t+1}$ is always in $H^{\perp}$. Thus, after performing a change of variable $\mathbf{v}_t := \boldsymbol{\Theta}_t + \mathbf{u}_t$ and exploiting the linearity of P_H, we obtain exactly FedPi :

$$\mathbf{v}_{t+1} = \mathbf{u}_t + 2\boldsymbol{\Theta}_{t+1} - \mathbf{z}_{t+1} = \mathbf{v}_t - \mathsf{P}_H\mathbf{v}_t + 2\mathsf{P}_H\mathsf{P}_{\mathsf{f}}^{\eta}\mathbf{v}_t - \mathsf{P}_{\mathsf{f}}^{\eta}\mathbf{v}_t = \tfrac{\mathbf{v}_t + \mathsf{R}_H\mathsf{R}_{\mathsf{f}}^{\eta}\mathbf{v}_t}{2}.$$

Comparing (2.5) and (2.9)-(2.10) it is clear that FedProx corresponds to fixing the dual variable $\mathbf{u}$ to the constant $\mathbf{0}$ in FedPi . We remark that step (2.9) is done at the clients' side while steps (2.10) and (2.11) are implemented at the server side. There is no communication overhead either, as the server need only communicate the sum $\boldsymbol{\Theta}_t + \mathbf{u}_t$ to the respective clients while the clients need only communicate their $\mathbf{z}_{t,i}$ to the server. The dual variable $\mathbf{u}$ is kept entirely at the server's expense.

Let us point out another subtle difference that may prove useful in FL: FedAvg and FedProx are inherently "synchronous" algorithms, in the sense that all participating clients start from a common, averaged model at the beginning of each communication round. In contrast, the local models $\mathbf{v}_t$ in FedPi may be different from each other, where we "correct" the common, average model $\boldsymbol{\Theta}_t$ with client-specific dual variables $\mathbf{u}_t$. This opens the possibility to personalization by designating the dual variable $\mathbf{u}$ in client-specific ways.

Lastly, we remark that in FedProx we need the step size η_t to diminish in order to converge to a solution of the original problem (2.1) whereas FedPi achieves the same with a constant step size η, although at the cost of doubling the memory cost at the server side.

[6]The acute readers may have recognized here the alternating direction method of multipliers (ADMM). Indeed, the equivalence of ADMM, Douglas-Rachford and partial inverse (under our FL setting) has long been known (e.g. Eckstein and Bertsekas 1992; Gabay 1983).

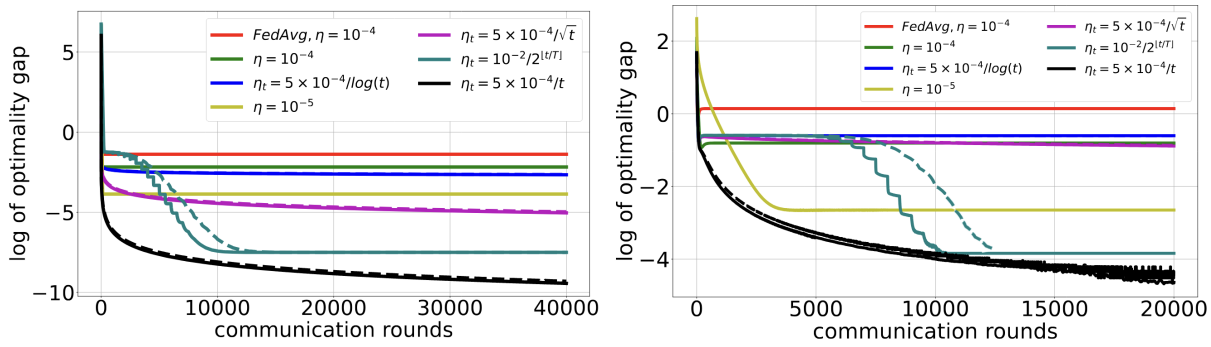


Figure 2.6: Effect of step size η and averaging on FedRP . Left: least squares; Right: logistic regression. The dashed and solid lines with the same color show the results obtained with and without the ergodic averaging step in Theorem 3, respectively. For exponentially decaying η_t, we use period T equal to 500 for both least squares and logistic regression.

2.4.5 FedRP as Reflection-Projection splitting

Examining the updates in (2.3), (2.5) and (2.7), we are naturally led to the following further variants (that have not been tried in FL to the best of our knowledge):

$$\boldsymbol{\Theta}_{t+1} = \mathsf{R}_H \mathsf{G}_{\mathsf{f}}^{\eta_t} \boldsymbol{\Theta}_t \tag{2.12}$$

$$\boldsymbol{\Theta}_{t+1} = \mathsf{R}_H \mathsf{P}_{\mathsf{f}}^{\eta_t} \boldsymbol{\Theta}_t \tag{2.13}$$

$$\text{FedRP} \ : \quad \boldsymbol{\Theta}_{t+1} = \mathsf{P}_H \mathsf{R}_{\mathsf{f}}^{\eta_t} \boldsymbol{\Theta}_t. \tag{2.14}$$

Interestingly, the last variant in (2.14), which we call FedRP, has been studied by Bauschke and Kruk (2004) under the assumption that $\mathsf{f} = \iota_{\mathsf{K}}$ is an indicator function of an *obtuse*[7] convex cone K. We prove the following result for FedRP:

Theorem 8. *Let each client participate in every round and the functions $\{f_i\}$ be convex. If the step size $\eta_t \equiv \eta$ is constant, then any fixed point of* FedRP *is a solution of the* regularized *problem* (2.6)*. If the reflector $\mathsf{R}_{\mathsf{f}}^{\eta}$ is idempotent (i.e. $\mathsf{R}_{\mathsf{f}}^{\eta}\mathsf{R}_{\mathsf{f}}^{\eta} = \mathsf{R}_{\mathsf{f}}^{\eta}$) and the clients are homogeneous, then the vanilla iterates $\boldsymbol{\Theta}_t$ of* FedRP *converge.*

It follows easily from a result in Bauschke and Kruk (2004) that a convex cone is obtuse iff its reflector is idempotent, and hence Theorem 8 strictly extends their result. We remark that Theorem 8 does not apply to the variants (2.12) and (2.13) since the reflector R_H is not idempotent (recall H is defined in (2.2)). Of course, we can prove (linear) convergence of both variants (2.12) and (2.13) if f is strongly convex. We omit the formal statement and proof since strong convexity does not appear to be easily satisfiable in practice.

In Figure 2.6, we show the effect of step size η on the convergence of FedRP on least squares and logistic regression. We run FedRP with both fixed and diminishing step sizes. In

[7]Recall that a convex cone K is obtuse iff its dual cone $\mathsf{K}^* := \{\boldsymbol{\theta}^* : \langle \boldsymbol{\theta}, \boldsymbol{\theta}^* \rangle \geq 0\}$ is contained in itself.

the experiments with diminishing step sizes, we set the initial value of η (i.e. η_0) to larger values–compared to the constant η values–to ensure that η does not get very small after the first few rounds. From the figure, one can see that FedRP with a fixed learning rate converges faster in early stage but to a sub-optimal solution. In contrast, FedRP with diminishing η converges slower early, but to better final solutions. It is interesting to note that the convergence behaviour of FedRP is similar to that of FedProx and when the conditions of Theorem 3 are satisfied, FedRP converges to a solution of the original problem (2.1), e.g. see the results for $\eta_t \propto 1/t$ which satisfies both conditions in Theorem 3. Moreover, one can see that ergodic averaging hardly affects the convergence of FedRP.

2.4.6 Effect of data heterogeneity

In Figure 2.7, we compare the performance of different splitting algorithms and how they respond to different degrees of data heterogeneity. We use least squares and a CNN model on MNIST for the convex and noconvex experiments, respectively. The details of the experimental setup are described in Section A.1 in the appendix.

In order to measure heterogeneity, we adopt the data heterogeneity measure of Khaled et al. (2020a), and quantify the amount of heterogeneity in clients' data for the least squares and logistic regression problems by

$$H := \frac{1}{n}\sum_{i=1}^{n} \| \nabla f_i(\boldsymbol{\theta}^*) \|_2^2,$$

where $\boldsymbol{\theta}^*$ is a minimizer of the original problem (2.1). When clients' data is homogeneous, all the local functions f_i have the same minimizer of $\boldsymbol{\theta}^*$ and $H = 0$. In general, the more heterogeneous the clients' data is, the larger H becomes.

As observed in Figure 2.7, for the convex setting, as expected, FedSplit , FedPi , and FedAvg with $k = 1$ achieve the smallest optimality gaps. The performance of all the algorithms deteriorates as clients' data become more heterogeneous (see Section A.1 in the appendix). For the nonconvex setting, the best results can be achieved by FedRP and FedAvg with a big k. It is noteworthy that in the nonconvex setting, the performance of FedAvg with $k = 1$ is significantly worse than that with $k = 100$.

2.5 Unification, implementation and acceleration

The operator splitting view of FL not only allows us to compare and understand the many existing FL algorithms but also opens the door for unifying and accelerating them. In this section we first introduce a grand scheme that unifies all aforementioned FL algorithms. Then, we provide two implementation variants that can accommodate different hardware and network constraints. Lastly, we explain how to adapt Anderson acceleration (Anderson 1965) to the unified FL algorithm, with almost no computation or communication overhead.

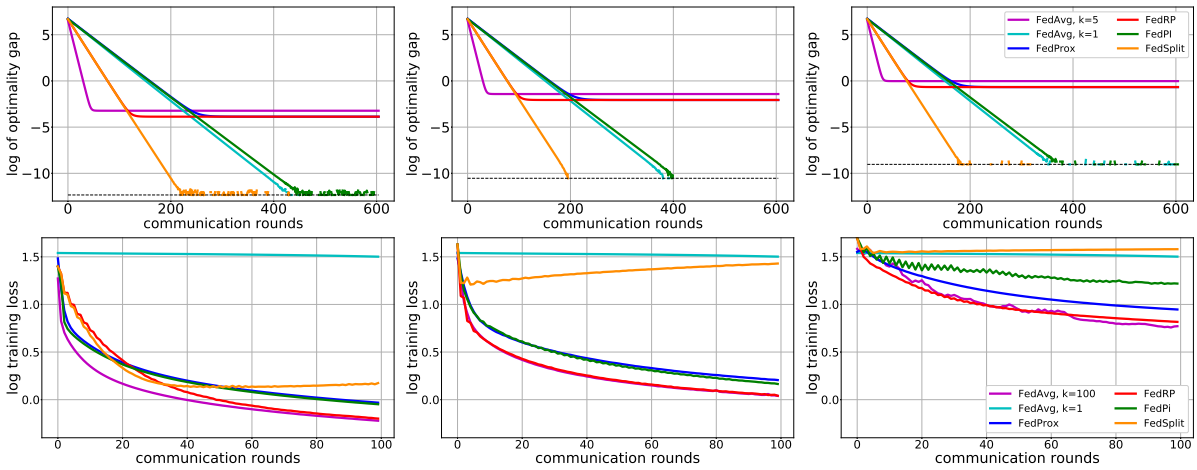


Figure 2.7: The effect of data heterogeneity on the performance of different splitting methods. The top row shows the results for the least squares, and the bottom row shows the results for nonconvex CNN model. Top-Left: small data heterogeneity with $H \approx 119 \times 10^3$; Top-Middle: moderately data heterogeneity with $H \approx 7.61 \times 10^6$; Top-Right: large data heterogeneity with $H \approx 190.3 \times 10^6$. Bottom-Left: i.i.d. data distribution; Bottom-Middle: non-i.i.d. data distribution with maximum 6 classes per client; Bottom-Right: non-i.i.d. data distribution with maximum 2 classes per client.

Unification. We introduce the following grand scheme:

$$\mathbf{z}_{t+1} = (1-\alpha_t)\mathbf{u}_t + \alpha_t \mathsf{P}_{\mathsf{f}}^{\eta_t}(\mathbf{u}_t) \tag{2.15}$$

$$\boldsymbol{\Theta}_{t+1} = (1-\beta_t)\mathbf{z}_{t+1} + \beta_t \mathsf{P}_H(\mathbf{z}_{t+1}) \tag{2.16}$$

$$\mathbf{u}_{t+1} = (1-\gamma_t)\mathbf{u}_t + \gamma_t \boldsymbol{\Theta}_{t+1}. \tag{2.17}$$

Table 2.1 confirms that the `FL` algorithms discussed in Section 2.4 are all special cases of this unifying scheme, which not only provides new (adaptive) variants but also clearly reveals the similarities and differences between seemingly different algorithms. It may thus be possible to transfer progress on one algorithm to the others and vice versa. We studied the effect of α, β and γ and found that γ mostly affects the convergence speed: the closer γ is to 1, the faster the convergence is, while α and β mostly determine the final optimality gap: the closer they are *both* to 2 (as in FedSplit and FedPi), the considerably smaller the final optimality gap is. However, setting only one of them close to 2 only has a minor effect on optimality gap or convergence speed.

Implementation. Interestingly, we also have two implementation possibilities:

- decentralized: In this variant, all the three updates in (2.15)-(2.17) are performed locally at each client. The server only acts as a bridge for synchronization by receiving $\mathbf{z}_{t+1,i}$ from the i-th client and returning the (same) averaged model $\mathsf{P}_H(\mathbf{z}_{t+1})$ to all clients. In a network where

Table 2.1: A unifying framework (2.15)-(2.17) for FL. Note that (a) FedAvg replaces the proximal update $\mathsf{P}_{\mathsf{f}}^{\eta_t}$ with a gradient update $\mathsf{G}_{\mathsf{f}}^{\eta_t}$; (b) ? indicates properties that remain to be studied; (c) "sampling" refers to selecting a subset of clients while "stochastic" refers to updating with stochastic gradient.

Algorithm	α	β	γ	$\eta_t \equiv \eta$	$\eta_t \to 0, \sum_t \eta_t = \infty$	nonconvex	sampling	stochastic
FedAvg	1	1	1	Eq. 2.1	Eq. 2.1	✓	✓	✓
FedProx	1	1	1	Eq. 2.6	Eq. 2.1	✓	✓	✓
FedSplit	2	2	1	Eq. 2.1	–	?	?	?
FedPi	2	2	$\frac{1}{2}$	Eq. 2.1	–	✓	?	?
FedRP	2	1	1	Eq. 2.6	Eq. 2.1	?	?	?

clients communicate directly with each other and compute $\mathsf{P}_H(\mathbf{z}_{t+1})$ locally, we may then dispense the server and become fully decentralized.

- centralized: In this variant, the i-th client only stores $\mathbf{u}_{t,i}$ and is responsible for computing and communicating $\mathsf{P}_{f_i}^{\eta_t}(\mathbf{u}_{t,i})$ to the server. In return, the server performs the rest of (2.15)-(2.17) and then sends (different) $\mathbf{u}_{t+1,i}$ back to the i-th client.

The two variants are of course mathematically equivalent in their basic form. However, as we see below, they lead to different acceleration techniques and we may prefer one over the other depending on the hardware and network constraints.

Acceleration. Let us abstract the grand scheme (2.15)-(2.17) as the map $\mathbf{u}_{t+1} = \mathsf{T}\mathbf{u}_t$, where T is nonexpansive if f is convex and $\alpha_t, \beta_t \in [0, 2], \gamma_t \in [0, 1]$. Following Fu et al. (2020), we may then apply the Anderson type-II acceleration to furthur improve convergence. Let $U = [\mathbf{u}_{t-\tau}, \ldots, \mathbf{u}_t]$ be given along with $T = [\mathsf{T}\mathbf{u}_{t-\tau}, \ldots, \mathsf{T}\mathbf{u}_t]$. We solve the following simple least-squares problem:

$$\boldsymbol{\pi}^* = \underset{\boldsymbol{\pi}^\top \mathbf{1}=1}{\operatorname{argmin}} \{\|(U - T)\boldsymbol{\pi}\|_2^2\} = \frac{G^\dagger \mathbf{1}}{\mathbf{1}^\top G^\dagger \mathbf{1}}, \qquad \text{where} \qquad G = (U - T)^\top (U - T).$$

(For simplicity we do not require $\boldsymbol{\pi}$ to be nonnegative.) Then, we update $\mathbf{u}_{t+1} = T\boldsymbol{\pi}^*$. Clearly, when $\tau = 0$, (trivially) $\pi^* = 1$ and we reduce to $\mathbf{u}_{t+1} = \mathsf{T}\mathbf{u}_t$. With a larger memory size τ, we may significantly improve convergence. Importantly, all heavy lifting (computation and storage) is done at the server side and we do not increase communication at all. We note that the same acceleration can also be applied on each client for computing $\mathsf{P}_{\mathsf{f}}^{\eta_t}$, if the clients can afford the memory cost.

We have seen how different variants (e.g. FedAvg , FedProx , FedSplit , FedPi , FedRP) compare to each other in §2.4. We performed further experiments to illustrate their behaviour under Anderson acceleration. (See Section A for details on our experimental setup.) As can be observed in Figure 2.8, Anderson acceleration helps FL algorithms converge considerably faster,

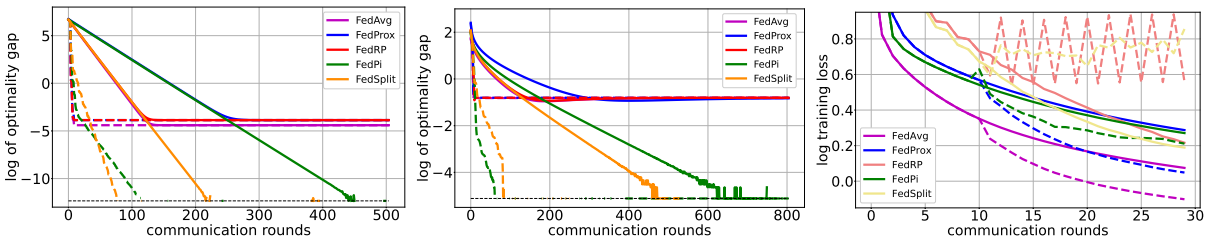


Figure 2.8: Effect of Anderson acceleration. Left: least squares with $\tau = 2$; Middle: logistic regression $\tau = 2$; Right: nonconvex CNN with $\tau = 10$. Dashed lines are the accelerated results.

all without incurring any overhead. For the convex models (least squares and logistic regression), our implementation of Anderson-acceleration speeds up all of the algorithms, especially FedAvg , FedProx and FedRP . However, for the nonconvex CNN model, it is beneficial only for FedAvg , FedProx and FedPi , while applying it to FedRP and FedSplit makes them unstable. It is noteworthy that we already know FedProx and FedPi are more stable than FedRP and FedSplit , respectively, and hence it makes intuitive sense that acceleration improves the two more stable algorithms. Another important point we wish to point out is that our acceleration method does not affect the quality of the algorithms' final solutions, but rather it just accelerates their convergence.

2.6 Summary

We have connected FL with the established theory of operator splitting from optimization, revealed new insights on existing algorithms and suggested new algorithmic variants and analysis. Our unified view makes it easy to understand, compare, implement and accelerate different FL algorithms in a streamlined and standardized fashion. Our experiments demonstrate some interesting differences in the convex and nonconvex settings, and in the early and late communication rounds. While our initial experiments confirmed the potential of Anderson-acceleration for FL, further work is required to formalize its effect in theory (such as (Zhang et al. 2020)) and to understand its relation with other momentum methods in FL. Studying the effect of stochasticity and extending our analysis to nonconvex functions is another direction for future studies.

Chapter 3

Noise-Aware Algorithm for Heterogeneous Differentially Private Federated Learning

3.1 Preface

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 most 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 sizes 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 a lower utility. We propose Robust-HDP, which efficiently estimates the true noise level in clients' model update 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 the proposed Robust-HDP.

3.2 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 (Geiping et al. 2020; Hitaj et al. 2017; Rigaki and García 2020; Wang et al. 2019; Zhu et al. 2019). Differential Privacy (Dwork 2011; Dwork et al. 2006a,b; Dwork and Roth 2014) has been used in DPFL systems to achieve such formal privacy guarantees. When there is a trusted server in the system, DP is provided by the server adding controlled noise to the aggregation of clients' updates

(Geyer et al. 2017; McMahan et al. 2018). When there is no trusted server, local perturbations, where each client randomizes its updates locally before sending them to the server (local DP), is also a solution (Zhao et al. 2021), while being limited in the sense that achieving privacy while preserving model utility is challenging, due to clients' independent local noise additions. Some solutions have been proposed for this, e.g., using a trusted shuffler system (Girgis et al. 2021; Liu et al. 2021b), 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 (Chathoth et al. 2022; Zhou et al. 2022), or suffer from suboptimal and vulnerable aggregation strategies on an untrusted server (based on clients' privacy parameters (Liu et al. 2021a)). We consider a heterogeneous DPFL systems with an *untrusted* server and propose an efficient aggregation strategy for the server, which is aware of the noise level in each client's model update: we propose to employ Robust PCA (RPCA) algorithm (Candès et al. 2009) by the curious 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 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 (§3.5.2).
- We propose Robust-HDP, a robust algorithm with a noise-aware aggregation strategy for heterogeneous DPFL (§3.5.4).
- 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 (§3.5.8).
- In various privacy heterogeneity scenarios across clients, we show that Robust-HDP improves utility and convergence speed compared to existing aggregation strategies (§3.6).

3.3 Related work

As explained in Chapter 1, there are various definitions of DP. In this chapter, we adopt the approximate $(\epsilon, \delta)-$DP definition (Definition 2), and use DPSGD algorithm for achieving it. 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.

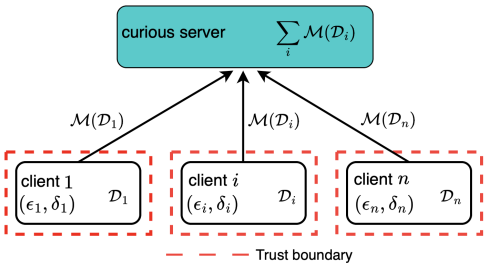


Figure 3.1: Security model in heterogeneous DPFL, where client i has local train data $\mathcal{D}_i$ and record-level privacy parameters (ϵ_i, δ_i), and does not trust any external parties.

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 previously used for centralized settings by Alaggan et al. 2017; Huang et al. 2020; Jorgensen et al. 2015; Kotsogiannis et al. 2020; Yu et al. 2023, followed by some recent works by Boenisch et al. 2023; Heo et al. 2023. Another similar work by 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.

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 1). Also, a work by 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. two FL systems differing by only one client's whole data) against membership inference attacks (Rigaki and García 2020; Wang et al. 2019). In contrast, we consider *trusted but curious* servers.

In the sense of Definition 2 of (ϵ, δ)-DP, the output of a randomized algorithm $\mathcal{M}$ is all the information that the untrusted server observes. We consider heterogeneous DPFL (Figure 3.1), where each client i has its own privacy parameters (ϵ_i, δ_i), and sends data-dependent computation results $\mathcal{M}(\mathcal{D}_i)$ (model updates) to the server. Also, in the context of Definition 2, the notion of neighboring datasets that we consider in this chapter is *record-level*: a pair of federated datasets $d = \{\mathcal{D}_1, \cdots, \mathcal{D}_n\}$ and $d' = \{\mathcal{D}'_1, \cdots, \mathcal{D}'_n\}$ differing in replacement of one data point in one client's dataset (e.g., $\mathcal{D}_i$ differs with $\mathcal{D}'_i$ in one data point, and $\mathcal{D}_j = \mathcal{D}'_j$ $(j \neq i)$).

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

Table 3.1: Features of different heterogeneous DPFL algorithms. $\times$: 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. 5**)	$w_i \propto \epsilon_i$	$\times$	$\times$	✓
PFA (**Liu et al. 2021a**)	$w_i \propto \epsilon_i$	$\times$	$\times$	$\times$
DPFedAvg (**Noble et al. 2021**)	$w_i \propto N_i$	✓	✓	✓
minimum ϵ (**Liu et al. 2021a**)	$w_i \propto N_i$	$\times$	✓	✓
Robust-HDP (**Alg. 1**)	$w_i \propto \frac{1}{\sigma_i^2}$	✓	✓	✓

larger than their true values) with the server. Also, the authors used aggregation strategy $w_i \propto \epsilon_i$ on server for PFA and another algorithm called WeiAvg (see Table 3.1). As we will show, *even if the server knows clients' true privacy parameters* $\{\epsilon_i\}_{i=1}^n$*, 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*.

The current state of the art in 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.

3.4 Notations

We consider an FL setting with n clients. Let $x \in \mathcal{X} \subseteq \mathbb{R}^d$ and $y \in \mathcal{Y} = \{1, \ldots, C\}$ denote an input data point and its target label. Client i holds dataset $\mathcal{D}_i = \{(x_{ij}, y_{ij})\}_{j=1}^{N_i}$ 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. 2017b, 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 $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n)$ 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 the 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. Table 3.2 summarizes the used notations.

3.5 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. At the t-th gradient update step on a current model $\boldsymbol{\theta}$, client i computes the

Table 3.2: Notations used in Chapter 3

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 communication rounds in the 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	used loss function, e.g., 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: $\texttt{Var}(\Delta\tilde{\boldsymbol{\theta}}_i^e \lvert \boldsymbol{\theta}^e)$

following noisy batch gradient by using DPSGD (Abadi et al. 2016):

$$\tilde{g}_i(\boldsymbol{\theta}) = \frac{1}{b_i}\Big[\Big(\sum_{j\in\mathcal{B}_i^t} \bar{g}_{ij}(\boldsymbol{\theta})\Big) + \mathcal{N}(\mathbf{0}, \sigma^2_{i,\texttt{DP}}\mathbb{I}_p)\Big], \tag{3.1}$$

where $\bar{g}_{ij}(\boldsymbol{\theta}) = \texttt{clip}(\nabla\ell(h(x_{ij}, \boldsymbol{\theta}), y_{ij}), c)$, and c is a clipping threshold. For a given vector $\mathbf{v}$, $\texttt{clip}(\mathbf{v}, c) = \min\{\|\mathbf{v}\|, c\} \cdot \frac{\mathbf{v}}{\|\mathbf{v}\|}$. Also, $\sigma_{i,\texttt{DP}} = c \cdot z(\epsilon_i, \delta_i, q_i, K_i, E)$: 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 $(\epsilon_i, \delta_i)-\texttt{DP}$ 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 $z(\epsilon_i, \delta_i, q_i, K_i, E)$, runs DPSGD locally for K_i epochs and sends the resulting noisy model updates to the server at the end of each round e. 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 3.5.1, which is used in Section 3.5.2, 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 client's model update, which leads to an efficient aggregation strategy in Section 3.5.4.

3.5.1 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' DP batch gradients. Depending on the value of clipping threshold c at the t-th gradient update step, we consider two general indicative cases:

1. $c < \|g_{ij}(\boldsymbol{\theta})\|, \forall j \in \mathcal{B}_i^t$**:** In this case, all sample gradients in the batch are clipped and from Equation (3.1), we get to the following expectation for the clipped batch gradient:

$$\mathbb{E}[\tilde{g}_i(\boldsymbol{\theta})] = \frac{1}{b_i}\sum_{j\in\mathcal{B}_i^t}\mathbb{E}[\bar{g}_{ij}(\boldsymbol{\theta})] = \frac{1}{b_i}\sum_{j\in\mathcal{B}_i^t} G_i(\boldsymbol{\theta}) = G_i(\boldsymbol{\theta}), \tag{3.2}$$

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 for all j and is denoted by $G_i(\boldsymbol{\theta})$. For an arbitrary random vector $\mathbf{v} = (v_1, \ldots, v_p)^\top \in \mathbb{R}^{p\times 1}$, we define $\texttt{Var}(\mathbf{v}) := \sum_{j=1}^p \mathbb{E}[(v_j - \mathbb{E}[v_j])^2]$, 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 (3.1), which is also random, can be computed as (see Appendix B for derivation):

$$\sigma_{i,\tilde{g}}^2 := \texttt{Var}(\tilde{g}_i(\boldsymbol{\theta})) = \frac{c^2 - \left\|G_i(\boldsymbol{\theta})\right\|^2}{b_i} + \frac{pc^2z^2(\epsilon_i,\delta_i,q_i,K_i,E)}{b_i^2} \approx \frac{pc^2z^2(\epsilon_i,\delta_i,q_i,K_i,E)}{b_i^2}, \tag{3.3}$$

where, the estimation is valid because $p \gg 1$. For instance, $p \approx 2\times 10^7$ for ResNet-34 for CIFAR100, and $c = 3$, which results in $pz^2(\epsilon_i,\delta_i,q_i,K_i,E)/b_i \gg 1$.

2. $c \geq \|g_{ij}(\boldsymbol{\theta})\|, \forall j \in \mathcal{B}_i^t$**:** 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 1). Hence:

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

$$\sigma_{i,\tilde{g}}^2 = \texttt{Var}(\tilde{g}_i(\boldsymbol{\theta})) = \texttt{Var}(g_i(\boldsymbol{\theta})) + \frac{p\sigma_{i,\texttt{DP}}^2}{b_i^2} \leq \sigma_{i,g}^2 + \frac{pc^2z^2(\epsilon_i,\delta_i,q_i,K_i,E)}{b_i^2}. \tag{3.4}$$

z is a sub-linearly increasing function of q_i and b_i (see Theorem 9 and Figure 3.2). 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).*

3.5.2 Noise level in clients' 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 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:

$$\Delta\tilde{\boldsymbol{\theta}}_i^e = \boldsymbol{\theta}_{i,E_i}^e - \boldsymbol{\theta}_{i,0}^e,$$

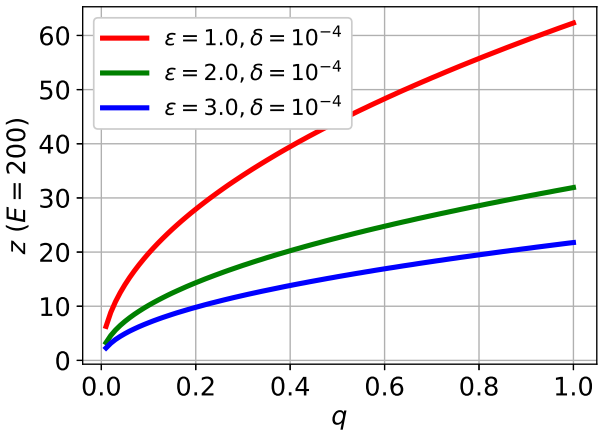


Figure 3.2: Plot of z v.s. q obtained from moments accountant (Abadi et al. 2016) in a centralized setting with $E = 200$. As observed, z increases sub-linearly with q (or equivalently with b).

$$\boldsymbol{\theta}^e_{i,k} = \boldsymbol{\theta}^e_{i,k-1} - \eta_l \tilde{g}_i(\boldsymbol{\theta}^e_{i,k-1}),\ k = 1, \ldots, E_i,$$

where $\boldsymbol{\theta}^e_{i,0} = \boldsymbol{\theta}^e$. 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, according to Equation (3.1). Hence:

$$\sigma_i^2 := \texttt{Var}(\Delta\tilde{\boldsymbol{\theta}}^e_i | \boldsymbol{\theta}^e) = K_i \cdot \lceil \frac{1}{q_i} \rceil \cdot \eta_l^2 \cdot \sigma^2_{i,\tilde{g}}, \tag{3.5}$$

where $\sigma^2_{i,\tilde{g}}$ was computed in Equation (3.3) and Equation (3.4) 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 3.3, left). However, note that N_i and q_i appear twice in Equation (3.5) with opposing effects. This makes the variation of σ_i^2 with N_i and q_i small (explained in details in Section F.3 in the Appendix). An important message of these important understandings is that ϵ_i *is not the only parameter of client* i *that determines* σ_i^2.

3.5.3 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:

$$\begin{aligned} \min_{w_i \geq 0} \quad & \texttt{Var}\Big(\sum_{i \in \mathcal{S}^e} w_i \Delta\tilde{\boldsymbol{\theta}}^e_i \big| \boldsymbol{\theta}^e\Big) = \sum_{i \in \mathcal{S}^e} {w_i}^2 \sigma_i^2, \\ & \texttt{s.t.} \sum_{i \in \mathcal{S}^e} w_i = 1, \end{aligned} \tag{3.6}$$

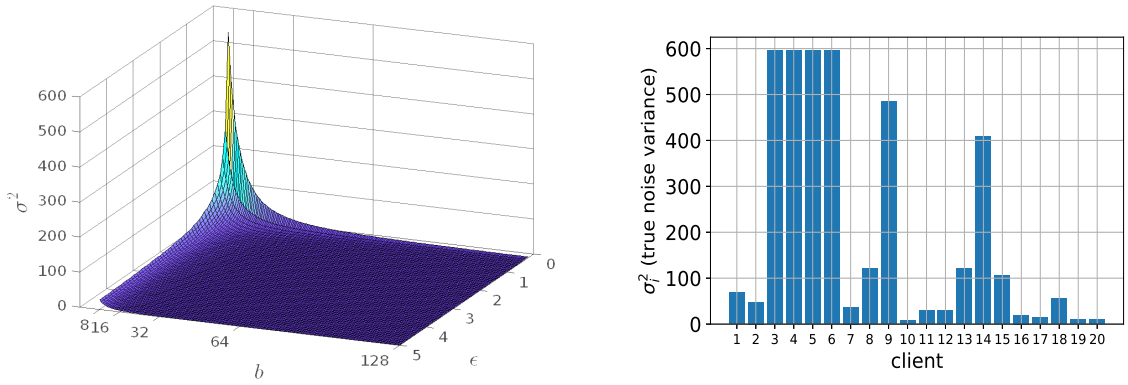


Figure 3.3: **Left:** 3D plot of noise variance σ_i^2 (Equation (3.5) with $K_i = 1, N_i = 2400, \eta_l = 0.01, c = 3, p = 28939$) based on b_i and the privacy budget ϵ_i. **Right:** the noise variances $\{\sigma_i^2\}_{i=1}^n$ in a DPFL system with $n = 20$ clients, where $\{(\epsilon_i, b_i)\}_{i=1}^n$ 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 corresponds to a point in the left figure.

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 and Kleinberg 2021; Fallah et al. 2023; Kang et al. 2023; Karimireddy et al. 2022) (as a larger ϵ_i usually 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 $\{w_i^*\}_{i=1}^n$.

3.5.4 Robust-HDP **algorithm**

Assuming n clients and full participation 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*

Algorithm 1: Robust-HDP

Input: Initial parameter $\boldsymbol{\theta}^0$, batch sizes $\{b_1, \ldots, b_n\}$, dataset sizes $\{N_1, \ldots, N_n\}$, noise scales $\{z_1, \ldots, z_n\}$, gradient norm bound c, local epochs $\{K_1, \ldots, K_n\}$, global round E, number of model parameters p, privacy accountant PA.

Output: $\boldsymbol{\theta}^E, \{\epsilon_1^E, \ldots, \epsilon_n^E\}$

Initialize $\boldsymbol{\theta}_0$ randomly

for $e \in [E]$ **do**

  sample a set of clients $\mathcal{S}^e \subseteq \{1, \ldots, n\}$

  for *each client* $i \in \mathcal{S}^e$ ***in parallel*** **do**

    $\Delta\tilde{\boldsymbol{\theta}}_i^e \leftarrow$DPSGD $(\boldsymbol{\theta}^e, b_i, N_i, K_i, z_i, c)$

    $\epsilon_i^e \leftarrow \mathsf{PA}(\frac{b_i}{N_i}, z_i, K_i, e)$

  $\mathbf{M} = [\Delta\tilde{\boldsymbol{\theta}}_1^e | \ldots | \Delta\tilde{\boldsymbol{\theta}}_{|\mathcal{S}^e|}^e] \in \mathbb{R}^{p \times |\mathcal{S}^e|}$

  $\mathbf{L}, \mathbf{S}$ = RPCA($\mathbf{M}$)

  for $i \in \mathcal{S}^e$ **do**

    $w_i^e \leftarrow \frac{1/\|\mathbf{S}_{:,i}\|_2^2}{\sum_{j \in \mathcal{S}^e} 1/\|\mathbf{S}_{:,j}\|_2^2}$

  $\boldsymbol{\theta}^{e+1} \leftarrow \boldsymbol{\theta}^e + \sum_{i \in \mathcal{S}^e} w_i^e \Delta\tilde{\boldsymbol{\theta}}_i^e$

Pursuit (Algorithm 6 in the appendix) without imposing much computational overhead to the server (Candès 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 (Candès 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 Section G in the Appendix for detailed discussion on data heterogeneity and further experiments). 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 (3.5)). Indeed, we observed such approximately sparse pattern for $\mathbf{S}$ in Figure 3.3 (right), where each barplot corresponds to the ℓ_2 norm of one column of $\mathbf{S}$. Thus, according to problem 3.6, 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.*

3.5.5 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 fewer rows:

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

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 $\mathbf{S}_j, j \in \{1, \ldots, Q\}$. 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.

3.5.6 Reliability of Robust-HDP

In order for Robust-HDP to assign the optimum aggregation weights $\{w_i^*\}$, *it suffices for it 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 $s_{i,j}'^2 \leq U$, where $\mathbb{E}[s'_{i,j}] = r s_{i,j}$, for some constant $r > 0$, and $\mathbb{E}[|s'_{i,j} - r s_{i,j}|^2] \leq \alpha_j^2$ (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:

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

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 (Candès 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 model update $\Delta\tilde{\boldsymbol{\theta}}_j^e$. See the values in Figure 3.3, right), a small deviation α_j^2 from $r^2\sigma_j^2$ still results in aggregation weights close to $\{w_i^*\}$.

3.5.7 Privacy analysis of Robust-HDP

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

Theorem 9. *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 *provides $(\epsilon_i, \delta_i)-DP$ to client 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. *Also, assuming clients' datasets are disjoint, 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 guarantees (ϵ_i, δ_i)-DP for client i, satisfying heterogeneous DPFL.

3.5.8 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 make the following assumptions about the clients' loss functions and then, we will provide the convergence analysis of the Robust-HDP algorithm.

Assumption 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 and $\|\nabla f_i(\boldsymbol{\theta}) - \nabla f_j(\boldsymbol{\theta})\| \leq \sigma$.*

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

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

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

Theorem 10 (Robust-HDP). *Assume that Assumptions 1 and 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:*

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

where $E_l^{min} = \min_i E_i$, i.e. the minimum number of local DPSGD steps per round across clients. Also, Ψ_p and Ψ_σ are two constants controlling the quality of the final model parameter returned by Robust-HDP *explained below. Also, assuming a large E, $\min_{0 \leq e \leq E-1} \mathbb{E}[\|\nabla f(\boldsymbol{\theta}^e)\|^2]$ happens close to the convergence time, i.e. at the last communication rounds.*

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 and Lan 2013):

$$\min_{0\leq e\leq E-1} \mathbb{E}[\|\nabla f(\boldsymbol{\theta}^e)\|^2] \leq \frac{12}{11E_l^{\min}-7}\frac{f(\boldsymbol{\theta}^0)-f^*}{E\eta_l} + \mathcal{O}(\eta_l),$$

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

3.6 Experiments

In this section, we evaluate the proposed Robust-HDP algorithm experimentally. See Appendix A for details of experimental setup and hyperparameter tuning used for evaluation of algorithms.

3.6.1 Experimental setup

Datasets, models and baseline algorithms: We evaluate our proposed method on four benchamrk datasets: MNIST (LeCun et al. 2010), FMNIST (Xiao et al. 2017) and CIFAR10/100 (Krizhevsky 2009) using CNN-based models. 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 A.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 3.3. 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 F and Appendix G.

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 3.3: Distributions of privacy parameters (ϵ), from which clients' privacy parameters are sampled.

3.6.2 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. 3.4, we have done a comparison in terms of the average test accuracy across clients. We observe that Robust-HDP outperforms the baselines. 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 clients' aggregated 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 3.5. 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 3.4), 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 Figure 3.6, 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, 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 3.6, right).

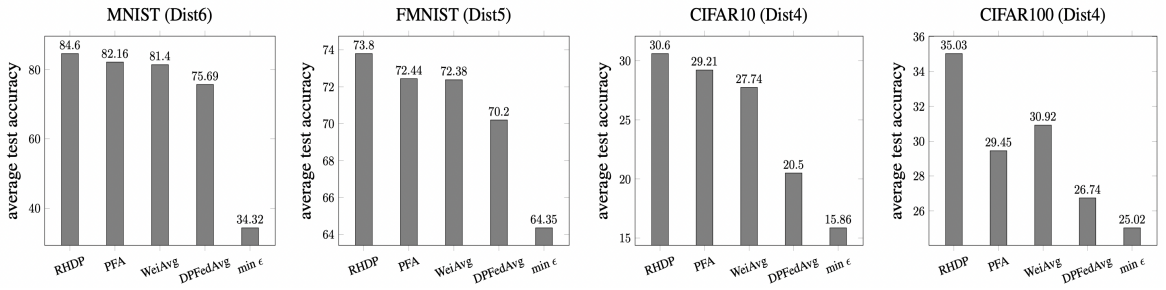


Figure 3.4: Comparison of average test accuracy between studied algorithms. See Tables B.7 to B.10 in the Appendix for detailed results.

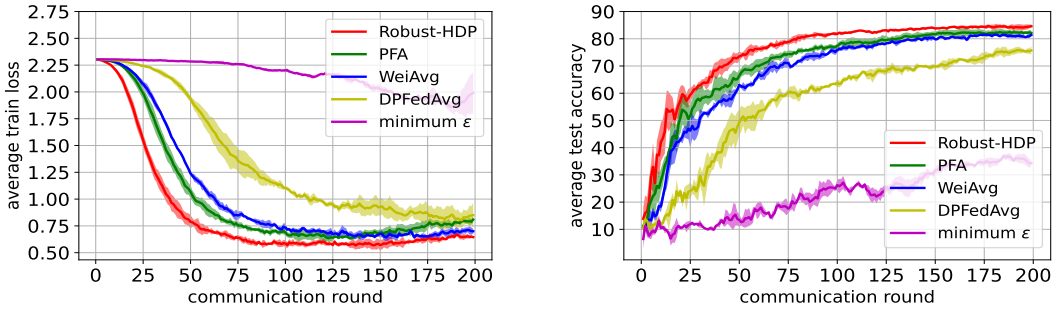


Figure 3.5: Convergence speed comparison on MNIST and Dist6. Minimum ϵ algorithm diverged in 1 out of 3 trials.

RQ4: How accurate Robust-HDP **is in estimating** $\{w_i^*\}$**?** Figure 3.7 left, compares the weight assignment of Robust-HDP with the optimum assignment (computed from Equation (3.6)) for CIFAR10 dataset and Dist2. As the model used for CIFAR10 is relatively large (with $p \approx 11\times10^6$), we have used the approximation method in Section 3.5.5 (with $Q = 1$ and $p' = 2\times10^5$). Figure 3.7 left, 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' in Section 3.5.5 on the precision of the aggregation weights returned by Robust-HDP. In Figure 3.7 right, 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 for estimating the noise variance in clients' model updates, hence more precise weight assignments. As explained in Section 3.5.5, when p is already large, we also avoid using too large values for p', as the main point of Section 3.5.5 was to pass 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.

RQ5: What is the effect of data heterogeneity 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

Table 3.4: The average *per parameter* noise variance (Equation (3.5) and Equation (3.3)) normalized by used learning rate ($\frac{\sum_{i=1}^{n} w_i^{e\,2}\sigma_i^2}{p\eta_t^2}$) 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.

alg \ dist	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 (Equation (3.6))	0.27	0.47	0.07	0.64	0.39	7.60	2.25	13.81	5.93

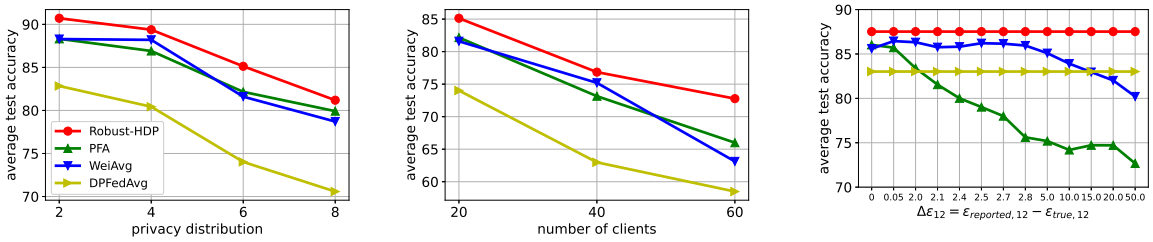


Figure 3.6: Performance comparison on MNIST. **Left:** effect of clients desired privacy on utility (detailed results in Table B.11 in the Appendix). **Middle:** effect of number of existing clients (privacy parameters of clients are sampled from Dist6) on utility (detailed results in Table B.12). **Right:** 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.

to have a useful RPCA decomposition $\mathbf{M} = [\Delta\tilde{\boldsymbol{\theta}}_1^e|\ldots|\Delta\tilde{\boldsymbol{\theta}}_n^e] = \mathbf{L} + \mathbf{S}$ at the end of the round, two conditions should be met (Candès 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 (Candès 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, given that the noise matrix $\mathbf{S}$ is sparse and there are large enough number of clients, we expect Robust-HDP to be successful in at least the decomposition task and the following noise estimation, even if the data split across clients is moderately heterogeneous.

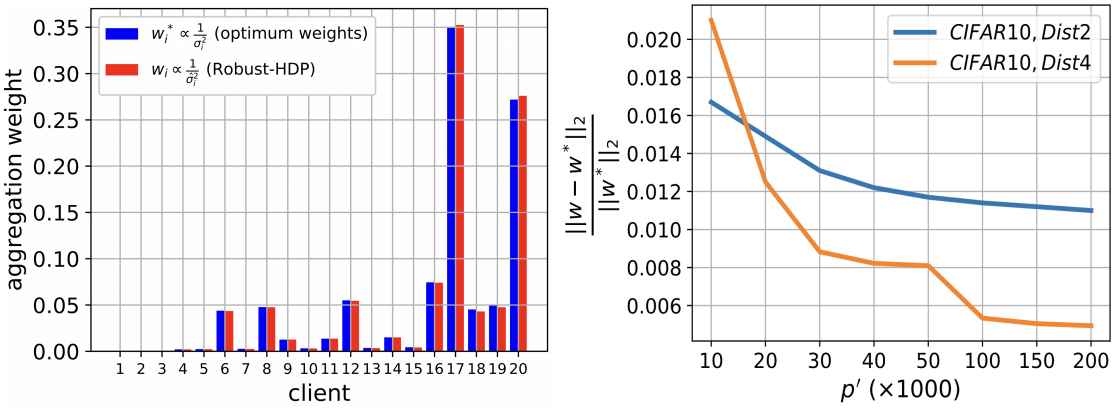


Figure 3.7: **Left:** Precision of Robust-HDP (red) compared to oracle optimum strategy (blue) for CIFAR10 and Dist2, when using the approximation method in Section 3.5.5 with $Q = 1$ and $p' = 2 \times 10^5$. **Right:** Effect of the parameter p', used in the approximation method explained in Section 3.5.5, on the precision of the weights returned by Robust-HDP for CIFAR10 (with $p \approx 11 \times 10^6$).

Table 3.5: 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**.

alg \ distr	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**

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 3.3, 3.4 and 3.5, 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 3.3), the better we can consider $\mathbf{S}$ as an approximately sparse matrix, which validates our RPCA decomposition.

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 3.5. As observed, Robust-HDP still outperforms the baselines in most of the cases. However, compared to the previous results, which were obtained for *i.i.d* data split, its superiority to the baseline algorithms has decreased. Detailed discussion of these results and scenarios with highly heterogeneous data splits are included in Appendix G.

3.7 Summary

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 aggregation strategy is robust, vastly applicable, scalable, and improves utility and convergence speed. It is also optimal in the sense that it minimizes the amount of noise in the aggregated model updates on the server.

Chapter 4

A Provably Robust Algorithm for Differentially Private Clustered Federated Learning

4.1 Preface

Federated learning (FL), which is a decentralized machine learning (ML) approach, often incorporates differential privacy (DP) to provide rigorous data privacy guarantees. Previous works attempted to address high *structured* data heterogeneity in vanilla FL settings through clustering clients (a.k.a clustered FL), but these methods remain sensitive and prone to errors, further exacerbated by the DP noise. This vulnerability makes the previous methods inappropriate for differentially private FL (DPFL) settings with structured data heterogeneity. To address this gap, we propose an algorithm for differentially private clustered FL, which is robust to the DP noise in the system and identifies the underlying clients' clusters correctly. To this end, we propose to cluster clients based on both their model updates and training loss values. Furthermore, for clustering clients' model updates at the end of the first round, our proposed approach addresses the server's uncertainties by employing large batch sizes as well as Gaussian Mixture Models (GMM) to reduce the impact of DP and stochastic noise and avoid potential clustering errors. This idea is efficient especially in privacy-sensitive scenarios with more DP noise, where the clustering task is more prone to errors. We provide theoretical analysis to justify our approach and evaluate it across diverse data distributions and privacy budgets. Our experimental results show its effectiveness in addressing large structured data heterogeneity in DPFL.

4.2 Introduction

Federated learning (FL) (McMahan et al. 2017a) is a collaborative ML paradigm, which allows multiple clients to train a shared global model without sharing their data. However, in order for

FL algorithms to ensure rigorous privacy guarantees against data privacy attacks (Geiping et al. 2020; Hitaj et al. 2017; Rigaki and García 2020; Wang et al. 2019; Zhu et al. 2019), they are reinforced with DP (Dwork 2011; Dwork et al. 2006a,b; Dwork and Roth 2014). This is done in the presence of a trusted server (Geyer et al. 2017; McMahan et al. 2018) as well as its absence (Duchi et al. 2013, 2018; Zhao et al. 2021). In the latter case and for sample-level DP, each client runs DPSGD (Abadi et al. 2016) locally and shares its noisy model updates with the server at the end of each round.

A key challenge in FL settings is ensuring an acceptable performance across clients under heterogeneous data distributions. Several existing works focus on accuracy parity across clients with a *single* common model by agnostic FL (Mohri et al. 2019) and client reweighting (Li et al. 2020a,d; Zhang et al. 2023). However, a single global model often fails to adapt to the data heterogeneity across clients (Dwork et al. 2012), especially when high data heterogeneity exists. Furthermore, when using a single model and augmenting FL with DP, different subgroups of clients are unevenly affected - even with loose privacy guarantees (Bagdasaryan and Shmatikov 2019; Farrand et al. 2020; Fioretto et al. 2022). In fact, subgroups with minority clients experience a larger drop in model utility, due to the inequitable gradient clipping in DPSGD (Abadi et al. 2016; Bagdasaryan and Shmatikov 2019; Esipova et al. 2022; Xu et al. 2021). Accordingly, some works proposed to use model personalization by multi-task learning (Li et al. 2020b; Marfoq et al. 2021; Smith et al. 2017; Wu et al. 2023), transfer learning (Li and Wang 2019; Liu et al. 2020) and clustered FL (Briggs et al. 2020; Ghosh et al. 2020; Mansour et al. 2020; Ruan and Joe-Wong 2021; Sattler et al. 2019; Werner et al. 2023). The latter has been proposed for vanilla FL and is suitable when “structured data heterogeneity" exists across clusters of clients (as in this work): subsets of clients can be naturally grouped together based on their data distributions and one model is learned for each group (cluster). However, as discussed in (Werner et al. 2023), the existing non-private clustered FL approaches are vulnerable to errors in clustering due to their sensitivity to: 1. model initialization 2. randomness in clients' model updates due to stochastic noise. The DP noise existing in DPFL systems' training mechanism exacerbates this vulnerability.

To address the aforementioned gap, we propose a differentially private clustered FL algorithm which uses both clients' model updates and loss values for clustering clients, making it more robust to DP/stochastic noise (Algorithm 2): 1) Justified by our theoretical analysis (Lemma 4 and 5) and in order to cluster clients correctly, our proposed algorithm uses a full batch size in the first FL round and a small batch size in the subsequent rounds, to reduce the noise in clients' model updates at the end of the first round. 2) Then, the server soft clusters clients based on these less noisy model updates using a Gaussian Mixture Model (GMM). Depending on the “confidence" of the learned GMM, the server keeps using it to soft cluster clients during the next few rounds (Section 4.5.5). 3) Finally, the server switches the clustering strategy to local clustering of clients based on their loss values in the remaining rounds. These altogether make our DP clustered FL algorithm effective and robust. The highlights of our contributions are as follows:

- We propose a DP clustered FL algorithm (R-DPCFL), which combines information from both clients' model updates and their loss values. The algorithm is robust and achieves high-quality clustering of clients, even in the presence of DP noise in the system (Algorithm 2).

- We theoretically prove that increasing clients' batch sizes in the first round (and decreasing them in the subsequent rounds) improves the server's ability to cluster clients based on their model updates at the end of the first round with high accuracy (Lemma 5).
- We show that utilizing sufficiently large client batch sizes in the first round (and sufficiently small batch sizes in the next rounds) enables super-linear convergence rate for learning a GMM on clients' model updates at the end of the first round. This leads to soft clustering of clients using a GMM with a low computational overhead (Theorem 11).
- We extensively evaluate across diverse datasets and scenarios, and demonstrate the effectiveness of our robust DP clustered FL algorithm in detecting the underlying cluster structure of clients, which leads to an overall utility improvement for the system (§4.6).

4.3 Related work

Model personalization is a technique for improving utility under *"moderate"* data heterogeneity (Li et al. 2020b; Liu et al. 2022), which usually leverages extra computations, e.g., extra local iterations (Li et al. 2020b). On the other hand, clustered FL has been proposed for personalized FL with *structured* data heterogeneity, where clients can be naturally partitioned into clusters: clients in the same cluster have similar data distributions, while there is a significant heterogeneity across different clusters. The ultimate goal in clustered FL is *to find the underlying cluster structure of clients* so that after learning one model for each cluster the overall performance for all clients will be maximized. This is not achievable by learning one global model for all clients, due to the existing data heterogeneity. Existing clustered FL algorithms group clients based on their loss values (Dwork et al. 2012; Ghosh et al. 2020; Liu et al. 2022; Mansour et al. 2020; Ruan and Joe-Wong 2021) or their model updates (based on e.g., their euclidean distance (Briggs et al. 2020; Werner et al. 2023) or cosine similarity (Sattler et al. 2019)). As shown by Werner et al. 2023, the algorithms are prone to clustering errors in the early rounds of FL training –due to gradient stochasticity, model initialization or the form of loss functions far from their optima– which can even propagate in the subsequent rounds. This vulnerability gets exacerbated by DP noise. Our proposed DP clustered FL algorithm is applicable to any setting characterized by a number of clients, where each client holds many data samples and needs sample-level privacy protection. Cross-silo FL systems can be considered as an instance. The closest study to this setting was recently done by Liu et al. 2022 studying the interplay between privacy and data heterogeneity in silo-specific sample-level DP. They show that when clients have large dataset sizes and "moderate" data heterogeneity: 1. participation in FL by clients is encouraged over local training, as the model averaging on the server side yields to mitigation of DP noise 2. under the same total privacy budget, model personalization - through mean regularized multi-task learning (MR-MTL) - leads to a better performance compared to training a single global model or local training by clients (see Section B.3 in the Appendix about MR-MTL formulation). Complementing the work, we show that MR-MTL, local training and even loss-based client clustering are not efficient for DPFL with "structured" data heterogeneity across clusters of clients.

4.4 Definitions, notations and assumptions

We keep using the approximate $(\epsilon, \delta)-\texttt{DP}$ (Definition 2) as the definition adopted by every client. We consider a DPFL system (see Figure 4.2, left), where there are n clients running DPSGD with the same "sample-level" privacy parameters (ϵ, δ): the set of information (including model updates and cluster selections) sent by client i to the server satisfies (ϵ, δ)-DP for all adjacent datasets $\mathcal{D}_i$ and $\mathcal{D}_i'$ of the client i differing in one sample (by replacement).

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 $\mathcal{D}_i$ with N_i samples from distribution $P_i(x, y) = P_i(y|x)P_i(x)$. Let $h : \mathcal{X} \times \boldsymbol{\theta} \to \mathbb{R}^C$ be the predictor function, which is parameterized by $\boldsymbol{\theta} \in \mathbb{R}^p$. Also, let $\ell : \mathbb{R}^C \times \mathcal{Y} \to \mathbb{R}_+$ be the used loss function (cross-entropy loss). Client i in the system has empirical train loss $f_i(\boldsymbol{\theta}) = \frac{1}{N_i} \sum_{(x,y)\in\mathcal{D}_i} [\ell(h(x, \boldsymbol{\theta}), y)]$, with minimum value f_i^*. There are E communication rounds indexed by e and K local epochs with learning rate η_l during each round. There are M clusters of clients indexed by m, and the server holds M cluster models $\{\boldsymbol{\theta}_m^e\}_{m=1}^M$ for them at the beginning of round e. Clients i and j belonging to the same cluster have the same data distributions, while there is a high data heterogeneity across clusters. $s(i)$ denotes the *unknown* true cluster of client i and $R^e(i)$ denotes the cluster assigned to it at the beginning of round e. Let us assume the batch size used by client i in the first round $e = 1$ is b_i^1, which may be different from the batch size $b_i^{>1}$ that it uses in the rest of the rounds $e > 1$. At the t-th gradient update during the round e, client i uses batch $\mathcal{B}_i^{e,t}$ with size b_i^e, and computes the following DP noisy batch gradient:

$$\tilde{g}_i^{e,t}(\boldsymbol{\theta}) = \frac{1}{b_i^e} \left[\Big(\sum_{j\in\mathcal{B}_i^{e,t}} \bar{g}_{ij}(\boldsymbol{\theta}) \Big) + \mathcal{N}(0, \sigma_{i,\texttt{DP}}^2 \mathbb{I}_p) \right], \tag{4.1}$$

where $\bar{g}_{ij}(\boldsymbol{\theta}) = \texttt{clip}(\nabla\ell(h(x_{ij}, \boldsymbol{\theta}), y_{ij}), c)$, and c is a clipping threshold to clip sample gradients: for a given vector $\mathbf{v}$, $\texttt{clip}(\mathbf{v}, c) = \min\{\|\mathbf{v}\|, c\} \cdot \frac{\mathbf{v}}{\|\mathbf{v}\|}$. Also, $\mathcal{N}$ is the Gaussian noise distribution with variance $\sigma_{i,\texttt{DP}}^2$, where $\sigma_{i,\texttt{DP}} = c \cdot z_i(\epsilon, \delta, b_i^1, b_i^{>1}, N_i, K, E)$, and z_i is the noise scale needed for achieving $(\epsilon, \delta)-\texttt{DP}$ by client i, which can be determined with a privacy accountant, e.g., Rényi-DP accountant (Mironov et al. 2019) used in this chapter. The accountant is capable of accounting composition of *heterogeneous* DP mechanisms (Mironov 2017). The privacy parameter δ is fixed to 10^{-4} in this section and for every client i: $\delta < N_i^{-1}$. Similar to Chapter 3, for an arbitrary random $\mathbf{v} = (v_1, \dots, v_p)^\top \in \mathbb{R}^{p\times 1}$, $\texttt{Var}(\mathbf{v}) := \sum_{j=1}^p \mathbb{E}[(v_j - \mathbb{E}[v_j])^2]$. Table 4.1 summarizes the used notations. Finally, we have the following assumption:

Assumption 3. *The stochastic gradient $g_i^{e,t}(\boldsymbol{\theta}) = \frac{1}{b_i^e} \sum_{j\in\mathcal{B}_i^{e,t}} g_{ij}(\boldsymbol{\theta})$ is an unbiased estimate of $\nabla f_i(\boldsymbol{\theta})$ with a bounded variance: $\forall \boldsymbol{\theta} \in \mathbb{R}^p : \texttt{Var}(g_i^{e,t}(\boldsymbol{\theta})) \leq \sigma_{i,g}^2(b_i^e)$. The tight bound $\sigma_{i,g}^2(b_i^e)$ is a constant depending only on the used batch size b_i^e: the larger the batch size b_i^e, the smaller $\sigma_{i,g}^2(b_i^e)$.*

Table 4.1: Used notations in chapter 4.

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
$\mathcal{D}_{i,aug}$	augmented local train set of client i
$\mathcal{B}_i^{e,t}$	the train data batch used by client i in round e and at the t-th gradient update
b_i^e	batch size of client i in round e: $\lvert\mathcal{B}_i^{e,t}\rvert = b_i^e$
b_i^1	batch size of client i during the first round $e = 1$
$b_i^{>1}$	set of batch sizes of client i during the rounds $e > 1$
ϵ, δ	desired DP privacy parameters
E	total number of communication rounds in the DPFL system, indexed by e
$\boldsymbol{\theta}_m^e$	model parameter for cluster m, at the beginning of global round e
K	number of local train epochs performed by clients during each global round e
η_l	the common learning rate used for DPSGD
h	predictor function, e.g., CNN model, with parameter $\boldsymbol{\theta}$
ℓ	cross entropy loss
$s(i)$	the true cluster of client i
$R^e(i)$	the cluster assigned to client i in round e
$\boldsymbol{\theta}_i^{e,0}$	the model parameter passed to client i at the beginning of round e to start its local training
$\Delta\tilde{\boldsymbol{\theta}}_i^e(b_i^e)$	the noisy update of client i at round e, starting from $\boldsymbol{\theta}_i^{e,0}$, with batch size b_i^e
$(\sigma_i^e(b_i^e))^2$	conditional variance of the update $\Delta\tilde{\boldsymbol{\theta}}_i^e$ of client i: $\mathtt{Var}(\Delta\tilde{\boldsymbol{\theta}}_i^e(b_i^e)\lvert\boldsymbol{\theta}_i^{e,0})$
$\mu_m^*(b^1)$	the center of the m-th cluster (when all use batch size b^1 in the first round)
$\Sigma_m^*(b^1)$	the covariance matrix of the m-th cluster (when all clients use batch size b^1 in the first round)
α_m^*	the prior probability of the m-th cluster

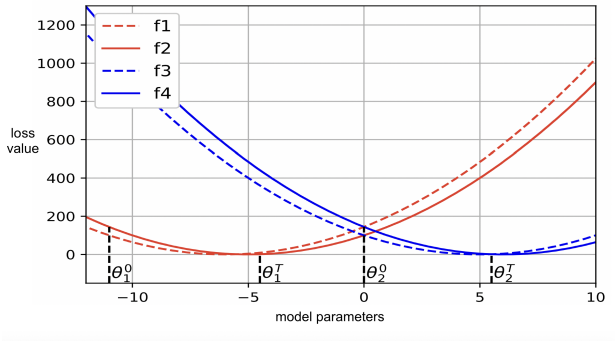


Figure 4.1: Loss-based clustering algorithms miscluster in the initial rounds, due to model initialization. Also, even with the assumption of perfect clustering of clients in the first rounds, clustering algorithms based on gradients (model updates) leads to clustering errors in the last rounds, due to the gradients approaching to zero.

4.5 Methodology and proposed algorithm

As discussed in (Werner et al. 2023), clustered FL algorithms which cluster clients based on their loss values (Ghosh et al. 2020; Mansour et al. 2020; Ruan and Joe-Wong 2021), i.e. assign client i to cluster $R^e(i) = \text{argmin}_m f_i(\boldsymbol{\theta}_m^e)$ at the beginning of round e, are prone to clustering errors in the first few rounds, mainly due to random initialization of cluster models $\{\boldsymbol{\theta}_m^0\}_{m=1}^M$. On the other hand, clustering clients based on their model updates (gradients) (Briggs et al. 2020; Sattler et al. 2019; Werner et al. 2023) makes sense only when the updates are obtained on the same model initialization. Additionally, even if we assume these algorithms can initially cluster clients perfectly in each round e, the clients' model updates (gradients) will approach zero as the clusters' model parameters converge. Hence, clients from different clusters may appear to belong to the same cluster, which results in clustering mistakes.

We now provide an example to elaborate on why clustering clients based on their losses (model updates) is prone to errors in the first (last) rounds. For example, consider Figure 4.1, where there are $M = 2$ clusters (red and blue) and $n = 4$ clients. The clients in the red cluster have loss functions $f_1(\theta) = 4(\theta + 6)^2$ and $f_2(\theta) = 4(\theta + 5)^2$ with optimum cluster parameter $\theta_1^\infty = -5.5$. Also, the the clients in the blue cluster have loss functions $f_3(\theta) = 4(\theta - 5)^2$ and $f_4(\theta) = 4(\theta - 6)^2$ with optimum cluster parameter $\theta_2^\infty = 5.5$. Clustering algorithms, which cluster clients based on their loss values on clusters' models, are vulnerable to model initialization. For example, in Figure 4.1, if we initialize the clusters' parameters with $\theta_1^0 = -11$ and $\theta_2^0 = 0$ (shown in the figure), all four clients will initially select cluster 2, since they have smaller losses on its parameter. At $\theta_2^0 = 0$, the average of clients' gradients (model updates) is zero, so all clients will remain stuck at θ_2^0 and will always select cluster 2.

On the other hand, clustering clients based on their model updates (gradients) (Briggs et al. 2020; Sattler et al. 2019; Werner et al. 2023) have clearly issues. One of these issues appears after some rounds of training. For instance, even if we assume these algorithms can initially cluster clients "perfectly" in each round e, the clients' model updates (gradients) will approach zero as the clusters' models converge to their optimum parameters. Hence, clients from different clusters may appear to belong to the same cluster, which results in clustering mistakes. For example, as shown in Figure 4.1, let us assume after T rounds of "correct" clustering of clients, the clusters' parameters get to $\theta_1^T = -4.5$ and $\theta_2^T = 5.5$ (shown in the figure). At these parameters, clients 1 and 2 (which have been "correctly" assigned to cluster 1 so far) will have gradients $f_1'(\theta_1^T) = 12$ and $f_2'(\theta_1^T) = 4$. Similarly, clients 3 and 4 (which have been "correctly" assigned to cluster 2 so far) will have $f_3'(\theta_2^T) = 4$ and $f_4'(\theta_2^T) = -4$. We see that f_2' is closer to f_3' and f_4' than to f_1', and in the next round it will be wrongly assigned to cluster 2. This happens while the clients are clearly distinguishable based on their losses, *as some progress in training has been made after T rounds*: $f_2(\theta_1^T) = 1$, while $f_2(\theta_2^T) = 21^2$, which clearly means that client 2 correctly belongs to cluster 1. Therefore, after making some progress in training the clusters' models, it makes more sense to use a loss-based clustering strategy than using a strategy based on clients' gradients (model updates).

Motivated by this vulnerability, *which will get exacerbated by DP noise*, we next propose a DP clustered FL algorithm which starts with clustering clients based on their model updates for the

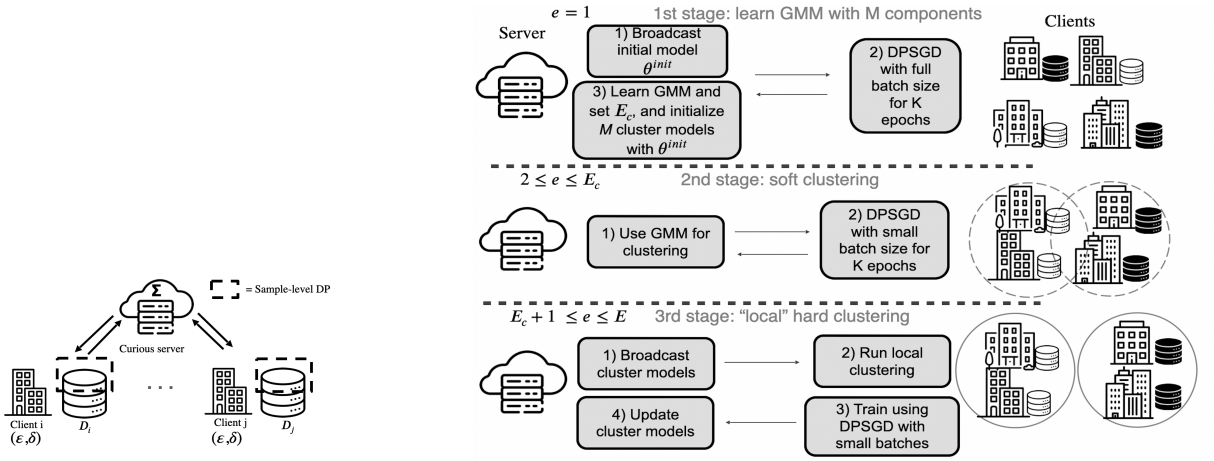


Figure 4.2: **Left:** Considered threat model in this work, where client i has local train data $\mathcal{D}_i$ and "sample-level" DP privacy parameters (ϵ, δ), and does not trust any external party. **Right:** Three main stages of the proposed R-DPCFL algorithm.

first several rounds and then switches its strategy to cluster clients based on their loss values. We augment this idea with some other non-obvious techniques to enhance the clustering accuracy.

4.5.1 R-DPCFL algorithm

R-DPCFL algorithm has three main steps (see Figure 4.2, right and Algorithm 2), as explained in the following:

1. In the first round, clients train the initial model $\boldsymbol{\theta}^{init}$ locally. They use full batch sizes in this round to make their model updates $\{\Delta\tilde{\boldsymbol{\theta}}_i^1\}_{i=1}^n$ less noisy [1]. Then, the server soft clusters them by learning GMM on their model updates. The number of clusters ($\mathbf{M}$) is either given or can be found by maximizing the confidence of the learned GMM (§4.5.5).

2. During the subsequent rounds $e \in \{2, \ldots, E_c\}$, the server uses the learned GMM to soft-cluster clients: client i uses a small batch size $b_i^{>1}$ and contributes to the training of each cluster (m) model proportional to the probability of its assignment to that cluster $(\pi_{i,m})$. The number of rounds E_c is set based on "confidence level" of the learned GMM (§4.5.5).

3. After the first E_c rounds, some progress has been made in the training of the cluster models $\{\boldsymbol{\theta}_m^{E_c}\}_{m=1}^M$. Now, clients' train loss values on cluster models are meaningful and is the right time to hard cluster clients based on their loss values during the remaining rounds to build more personalized models per cluster: $R^e(i) = \text{argmin}_m\, f_i(\boldsymbol{\theta}_m^e)$.

In §4.5.2 and §4.5.3, we provide theoretical justifications for the proposed method and why using a full batch size in the first round and a small batch size in the subsequent rounds helps.

[1] Note that even when clients have a limited memory budget, they can still perform DPSGD with full batch size and no computational overhead by using gradient accumulation technique (see Appendix G of Chapter 3)

Algorithm 2: R-DPCFL

Input: Initial parameter $\boldsymbol{\theta}^{init}$, dataset sizes $\{N_1, \ldots, N_n\}$, batch sizes $\{b_1^{>1}, \ldots, b_n^{>1}\}$, clip bound c, local epochs K, global round E, number of clusters M (optional)

Output: cluster models $\{\boldsymbol{\theta}_m^E\}_{m=1}^M$

for *each client* $i \in \{1, \ldots, n\}$ **do**
  $b_i^1 \leftarrow N_i$; // full batch size
  $z_i \leftarrow \texttt{RDP}(\epsilon, \delta, b_i^1, b_i^{>1}, N_i, K, E)$
for $e \in \{1, \ldots, E\}$ **do**
  if $e = 1$ **then**
    for *each client* $i \in \{1, \ldots, n\}$ ***in parallel*** **do**
      $\Delta\tilde{\boldsymbol{\theta}}_i^1 \leftarrow \texttt{DPSGD}\ (\boldsymbol{\theta}^{init}, b_i^1, N_i, K, z_i, c)$
    on server:
    if M *is unknown* **then**
      $M = \text{argmax}_{M'}\, \texttt{MSS}\Big(\texttt{GMM}(\Delta\tilde{\boldsymbol{\theta}}_1^1, \ldots, \Delta\tilde{\boldsymbol{\theta}}_n^1; M')\Big)$; // (Sec. 4.5.5)
    $\{\pi_1, \ldots, \pi_n, \texttt{MPO}\} = \texttt{GMM}(\Delta\tilde{\boldsymbol{\theta}}_1^1, \ldots, \Delta\tilde{\boldsymbol{\theta}}_n^1; M)$; // **1st stage:** GMM
    set $E_c(\texttt{MPO})$; // (Sec. 4.5.5)
    Initialize cluster models uniformly: $\boldsymbol{\theta}_1^2 = \ldots = \boldsymbol{\theta}_M^2 = \boldsymbol{\theta}^{init}$
    continue ; // go to round $e = 2$
  else if $e \in \{2, \ldots, E_c\}$ **then**
    for *each client* $i \in \{1, \ldots, n\}$ **do**
      $R^e(i) \leftarrow m$ *with probability* $\pi_i[m]$; // **2nd stage:** soft clustering
  else
    on server: broadcast cluster models $\{\boldsymbol{\theta}_m^e\}_{m=1}^M$ to all clients
    for *each client* $i \in \{1, \ldots, n\}$ **do**
      $R^e(i) = \text{argmin}_m\, f_i(\boldsymbol{\theta}_m^e)$; // **3rd stage:** private clustering
  for *each client* $i \in \{1, .., n\}$ ***in parallel*** **do**
    $\Delta\tilde{\boldsymbol{\theta}}_i^e \leftarrow \texttt{DPSGD}\ (\boldsymbol{\theta}_{R^e(i)}^e, b_i^{>1}, N_i, K, z_i, c)$; // batch size $b_i^{>1}$
  on server:
  for *each client* $i \in \{1, \ldots, n\}$ **do**
    $w_i^e \leftarrow \frac{1}{\sum_{j=1}^n \mathbb{1}_{R^e(j)=R^e(i)}}$
  for $m \in \{1, \ldots, M\}$ **do**
    $\boldsymbol{\theta}_m^{e+1} \leftarrow \boldsymbol{\theta}_m^e + \sum_{i \in \{1,\ldots,n\}} \mathbb{1}_{R^e(i)=m} w_i^e \Delta\tilde{\boldsymbol{\theta}}_i^e$

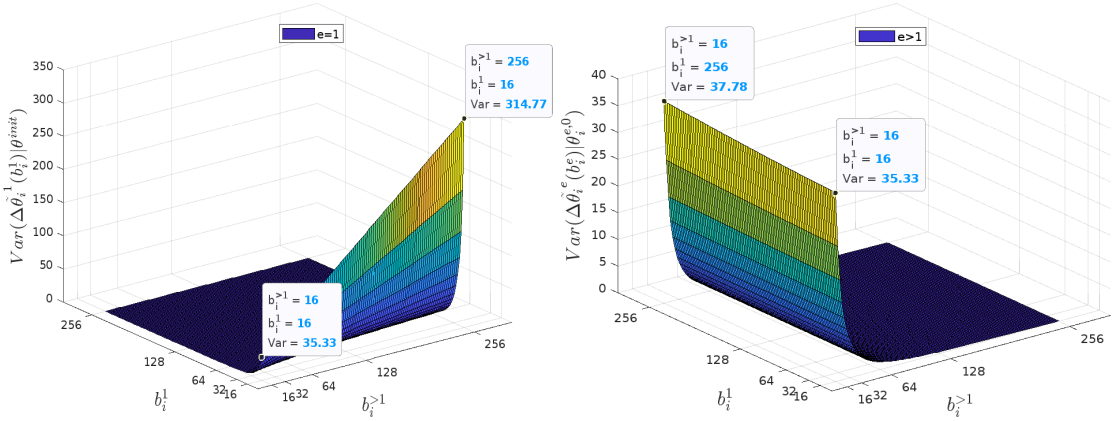


Figure 4.3: Plot of $\texttt{Var}(\Delta\tilde{\boldsymbol{\theta}}_i^1(b_i^1)|\boldsymbol{\theta}_i^{init})$ **(left)** and $\texttt{Var}(\Delta\tilde{\boldsymbol{\theta}}_i^e(b_i^e)|\boldsymbol{\theta}_i^{e,0})$ $(e > 1)$ **(right)** v.s. both b_i^1 and $b_i^{>1}$. There are two clear takeaways: 1) for all $e \in \{1, \cdots, E\}$, $\texttt{Var}(\Delta\tilde{\boldsymbol{\theta}}_i^e(b_i^e)|\boldsymbol{\theta}_i^{e,0})$ decreases with b_i^e steeply (from Lemma 4). 2) The effect of $b_i^{>1}$ on $\texttt{Var}(\Delta\tilde{\boldsymbol{\theta}}_i^1(b_i^1)|\boldsymbol{\theta}_i^{init})$ (left figure) is considerable. The reason is that $b_i^{>1}$ is used in $E-1$ rounds and affects the noise scale $z_i(\epsilon, \delta, b_i^1, b_i^{>1}, N_i, K, E)$ used by DPSGD: **see Figure 4.4 for the plot of** $z_i(\epsilon, \delta, b_i^1, b_i^{>1}, N_i, K, E)$ **v.s.** b_i^1 **and** $b_i^{>1}$. The results are obtained on CIFAR10 from Rényi-DP Accountant (Mironov et al. 2019) in a setting with $N_i = 6600, \epsilon = 5, \delta = 10^{-4}, c = 3, K = 1, E = 200, p = 11,181,642, \eta_l = 5 \times 10^{-4}$.

4.5.2 Reducing GMM uncertainty via using full batch sizes in the first round and small batch sizes in the subsequent rounds

The DP noise in $\{\Delta\tilde{\boldsymbol{\theta}}_i^1\}_{i=1}^n$ makes it harder for the server to cluster clients by learning a GMM on the model updates. The following lemma quantifies this noise amount by extending the results in Section 3.5.2 of Chapter 3 to when different batch sizes are used in the first and the next rounds:

Lemma 4. *Let us assume $\boldsymbol{\theta}_i^{e,0}$ is the model parameter passed to client i at the beginning of round e. After K local epochs with step size η_l, the client generates the noisy DP model update $\Delta\tilde{\boldsymbol{\theta}}_i^e(b_i^e)$ at the end of the round. The amount of noise in the resulting model update can be quantified as:*

$$(\sigma_i^e(b_i^e))^2 := \texttt{Var}(\Delta\tilde{\boldsymbol{\theta}}_i^e(b_i^e)|\boldsymbol{\theta}_i^{e,0}) \approx K \cdot N_i \cdot \eta_l^2 \cdot \frac{pc^2 z_i^2(\epsilon, \delta, b_i^1, b_i^{>1}, N_i, K, E)}{(b_i^e)^3}.$$

The first conclusion from the lemma is that the noise level in $\Delta\tilde{\boldsymbol{\theta}}_i^e$ rapidly declines as b_i^e increases: See Figure 4.3 and the effect of batch size b_i^1 on $\texttt{Var}(\Delta\tilde{\boldsymbol{\theta}}_i^1|\boldsymbol{\theta}_i^{init})$ (on the left) and the effect of batch size $b_i^{>1}$ on $\texttt{Var}(\Delta\tilde{\boldsymbol{\theta}}_i^e|\boldsymbol{\theta}_i^{e,0})$ $(e > 1)$ (on the right). Let us consider $e = 1$ especially: If all clients use full batch sizes in the first round (i.e. $b_i^1 = N_i$ for every client i), it becomes much easier for the server to cluster them at the end of the first round by learning a GMM on $\{\Delta\tilde{\boldsymbol{\theta}}_i^1\}_{i=1}^n$, as their updates become more separable. An illustration of this is shown in Figure 4.5. In the

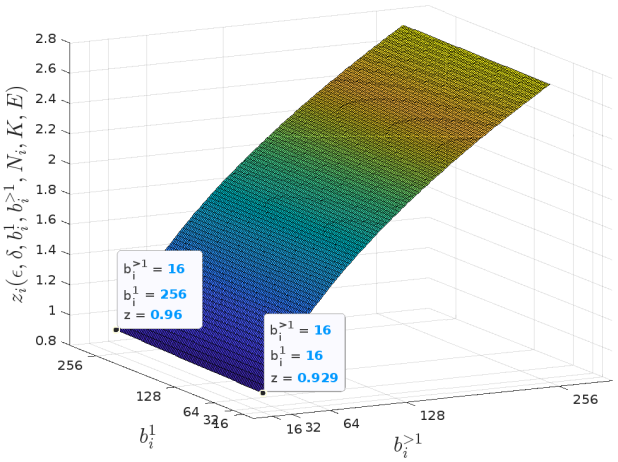


Figure 4.4: Plot of $z_i(\epsilon, \delta, b_i^1, b_i^{>1}, N_i, K, E)$ v.s. b_i^1 and $b_i^{>1}$ obtained from Rényi-DP Accountant (Mironov et al. 2019) in a setting with $N_i = 6600, \epsilon = 5, \delta = 10^{-4}, K = 1, E = 200$. It is clearly observed that the effect of $b_i^{>1}$ is much more than the effect of b_i^1. The reason is that $b_i^{>1}$ is used in $E - 1$ rounds, while b_i^1 is used only in the first round. So the value of $b_i^{>1}$ affects z_i the most.

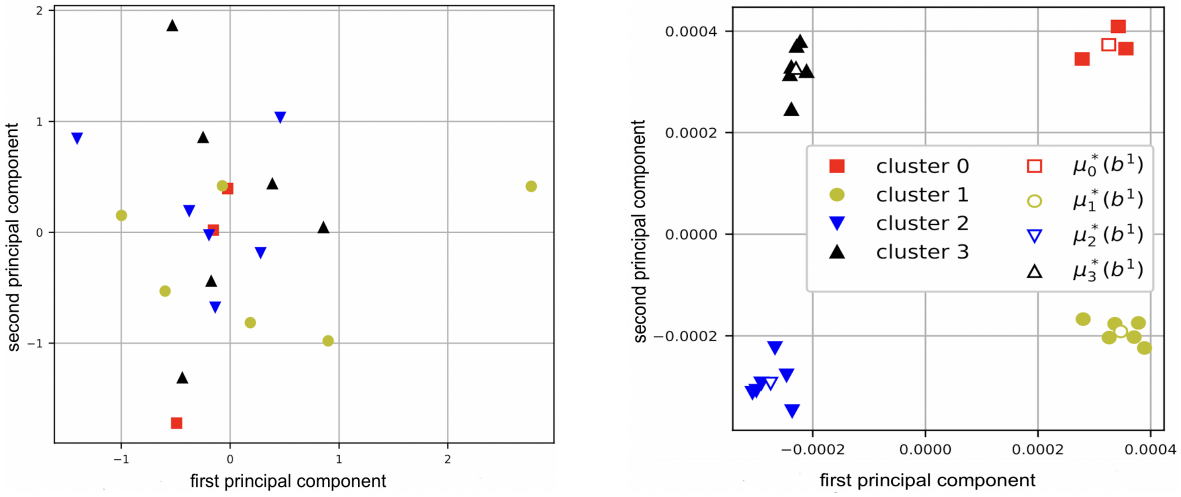


Figure 4.5: PCA visualization of updates $\{\Delta\tilde{\boldsymbol{\theta}}_i^1\}_{i=1}^n$ on 2D space. **Left:** $\epsilon_i = 10$, $b_i^e = 32$ for all i and e. **Right:** $\epsilon_i = 10$, $b_i^1 = b^1 = N = 6600$, **i.e. full batch sizes** (assuming $N_i = N = 6600$ for all clients), and $b_i^{>1} = 32$ for all i. $\mu_m^*(b^1)$ shows the center of cluster m, when all clients use batch size b^1 during the first round. The model updates are obtained from clients running DPSGD for $K = 1$ epochs locally on CIFAR10 with covariate shift (rotation) across clusters, and under the same values as in Figure 4.3.

next section, we will provide a theoretical justification for this observation. As the second key takeaway, Figure 4.3, left shows that **in order to make** $\{\Delta\tilde{\boldsymbol{\theta}}_i^1\}_{i=1}^n$ **less noisy, we have to make** $\{b_i^1\}_{i=1}^n$ **as large as possible and also keep** $\{b_i^{>1}\}_{i=1}^n$ **small**[2].

[2]In fact, there is a close relation between the result of Lemma 4 and the law of large numbers. See Appendix F for more details.

4.5.3 Effect of batch sizes $\{b_i^1\}_{i=1}^n$ on the separation between clusters

In this section, we provide a theoretical justification for the observation in Figure 4.5, right. *For simplicity, let us assume clients have the same dataset sizes and first batch sizes:* $\forall i : N_i = N, b_i^1 = b^1$. Also, remember that $\boldsymbol{\theta}_i^{1,0} = \boldsymbol{\theta}^{init}$. Having uniform privacy parameters (ϵ, δ), we have: $\forall i : (\sigma_i^1(b^1))^2 := \texttt{Var}[\Delta\tilde{\boldsymbol{\theta}}_i^1(b^1)|\boldsymbol{\theta}^{init}] = (\sigma^1(b^1))^2$. Hence, we can consider the model updates $\{\Delta\tilde{\boldsymbol{\theta}}_i^1(b^1)\}_{i=1}^n$ as the samples from a mixture of M Gaussians with mean, covariance matrix, prior probability parameters: $\psi^*(b^1) = \{\mu_m^*(b^1), \Sigma_m^*(b^1), \alpha_m^*\}_{m=1}^M$, where $\forall m : \alpha_m^* > 0$ and $\mu_m^*(b^1) \neq \mu_{m'}^*(b^1)$ $(m \neq m')$, due to data heterogeneity:

$$\mu_m^*(b^1) := \mathbb{E}\Big[\Delta\tilde{\boldsymbol{\theta}}_i^1(b_i^1)\Big|\boldsymbol{\theta}^{init}, b_i^1 = b^1, s(i) = m\Big],$$

$$\Sigma_m^*(b^1) := \mathbb{E}\Big[\big(\Delta\tilde{\boldsymbol{\theta}}_i^1(b_i^1) - \mu_m^*(b^1)\big)\big(\Delta\tilde{\boldsymbol{\theta}}_i^1(b_i^1) - \mu_m^*(b^1)\big)^\top\Big|\boldsymbol{\theta}^{init}, b_i^1 = b^1, s(i) = m\Big] = \frac{(\sigma^1(b^1))^2}{p}\mathbb{I}_p, \tag{4.2}$$

where the last equality is from $\texttt{Var}[\Delta\tilde{\boldsymbol{\theta}}_i^1|\boldsymbol{\theta}^{init}, b_i^1 = b^1] = \mathbb{E}[\|\Delta\tilde{\boldsymbol{\theta}}_i^1 - \mu_{s(i)}^*(b^1)\|^2] = (\sigma^1(b^1))^2$ and that the noises existing in each of the p elements of $\Delta\tilde{\boldsymbol{\theta}}_i^1$ are *i.i.d* (hence, $\Sigma_m^*(b^1)$ is a diagonal covariance matrix with equal diagonal elements). Intuitively, we expect more separation between the true Gaussian components $\{\mathcal{N}\big(\mu_m^*(b^1), \Sigma_m^*(b^1)\big)\}_{m=1}^M$, from which clients' updates $\{\Delta\tilde{\boldsymbol{\theta}}_i^1\}_{i=1}^n$ are sampled, to make the model updates more distinguishable for the server. Next, we show that the overlap between the Gaussian components $\{\mathcal{N}(\mu_m^*(b^1), \Sigma_m^*(b^1))\}_{m=1}^M$ decreases *fast* with b^1:

Lemma 5. *Let* $\Delta_{m,m'}(b^1) := \|\mu_m^*(b^1) - \mu_{m'}^*(b^1)\|$ *when* $\forall i : b_i^1 = b^1$. *The overlap between components* $\mathcal{N}\big(\mu_m^*(b^1), \Sigma_m^*(b^1)\big)$ *and* $\mathcal{N}\big(\mu_{m'}^*(b^1), \Sigma_{m'}^*(b^1)\big)$ *is* $O_{m,m'} = 2Q(\frac{\sqrt{p}\Delta_{m,m'}(b^1)}{2\sigma^1(b^1)})$, *where* $(\sigma^1(b^1))^2 := \texttt{Var}[\Delta\tilde{\boldsymbol{\theta}}_i^1(b_i^1)|\boldsymbol{\theta}^{init}, b_i^1 = b^1]$ *and* $Q(\cdot)$ *is the Q function. Furthermore, if we increase* $b_i^1 = b^1$ *to* $b_i^1 = kb^1 \leq N$ *(for all* i*), we have* $O_{m,m'} \leq 2Q(\frac{\sqrt{kp}\Delta_{m,m'}(b^1)}{2\rho\sigma^1(b^1)})$, *where* $1 \leq \rho \in \mathcal{O}(1)$ *is a small constant.*

Note that for a batch size b^1, the terms $\Delta_{m,m'}(b^1)$ and $\sigma^1(b^1)$ represent the "data heterogeneity level across clusters m and m'" and "privacy sensitivity of their clients", respectively. We define their "separation score" as $\texttt{SS}(m, m') := \frac{\sqrt{p}\Delta_{m,m'}(b^1)}{2\sigma^1(b^1)} = \frac{\Delta_{m,m'}(b^1)}{2\sigma^1(b^1)/\sqrt{p}}$. The larger separation score $\texttt{SS}(m, m')$, the smaller their pairwise overlap $O_{m,m'} = 2Q(\texttt{SS}(m, m'))$. Based on the form of the Q function, an $\texttt{SS}(m, m')$ above 3 can be considered as a complete separation.

4.5.4 Convergence rate of EM for learning GMM

Let us define the maximum pairwise overlap between the true M Gaussian components in $\psi^*(b^1) = \{\mu_m^*(b^1), \Sigma_m^*(b^1), \alpha_m^*\}_{m=1}^M$, as $O^{\max}(\psi^*(b^1)) = \max_{m,m'} O_{m,m'}(\psi^*(b^1))$. According to

Lemma 5, when b^1 is large enough, $O^{\texttt{max}}(\psi^*(b^1))$ decreases (like in Figure 4.5, right) and we can expect EM to converge to the true GMM parameters $\psi^*(b^1)$. Next, we analyze the local convergence rate of EM around $\psi^*(b^1)$.

Theorem 11. *(Ma et al. 2000) Given model updates $\{\Delta\tilde{\boldsymbol{\theta}}_i^1(b^1)\}_{i=1}^n$, as samples from a true mixture of Gaussians $\psi^*(b^1) = \{\mathcal{N}\big(\mu_m^*(b^1), \Sigma_m^*(b^1)\big), \alpha_m^*\}_{m=1}^M$, if $O^{max}(\psi^*(b^1))$ is small enough, then:*

$$\lim_{r\to\infty} \frac{\|\psi^{r+1} - \psi^*(b^1)\|}{\|\psi^r - \psi^*(b^1)\|} = o\Big(\big[O^{max}(\psi^*(b^1))\big]^{0.5-\gamma}\Big),$$

as n increases. ψ^r is the parameters returned by EM after r iterations. γ is an arbitrary small positive number, and $o(x)$ means it is a higher order infinitesimal as $x \to 0 : \lim_{x\to 0} \frac{o(x)}{x} = 0$.

This means that convergence rate of EM around the true solution $\psi^*(b^1)$ is faster than how $\big[O^{\texttt{max}}(\psi^*(b^1))\big]^{0.5-\gamma}$ decreases with b^1 (from Lemma 5). *Hence, the computational complexity of learning the GMM in the first round also decreases fast as b^1 increases.*

4.5.5 Applicability of R-DPCFL

As we observed in Lemma 5, the separation score $\texttt{SS}(m, m')$ (the overlap $O_{m,m'}$) increases (decreases) as b^1 increases. Remember that $\texttt{SS}(m, m') = \frac{\Delta_{m,m'}(b^1)}{2\sigma^1(b^1)/\sqrt{p}}$, and note that $(\sigma^1(b^1))^2/p$ is the value of diagonal elements of covariance matrices of Gaussian components, which the GMM aims to learn (see Equation (4.2)). Therefore, when the GMM is learned, we can use its parameters to get an estimate score $\hat{\texttt{SS}}(m, m')$ for every pair of clusters m and m'. Then, we can define the "minimum pairwise separation score" as $\texttt{MSS}(\epsilon, \delta, \{b_i^1\}_{i=1}^n, \{b_i^{>1}\}_{i=1}^n) = \min_{m,m'} \hat{\texttt{SS}}(m, m') \in [0, +\infty)$ as a **measure of confidence** of the learned GMM in its identified clusters. The larger the MSS of a learned GMM, the more "confident" it is in its clustering decisions. For instance, if we learn a GMM on Figure 4.5 left, it will have a much smaller MSS than when we learn a GMM on Figure 4.5 right. We can similarly define the estimated "maximum pairwise overlap" for a learned GMM as $\texttt{MPO} = 2Q(\texttt{MSS}) \in [0, 1)$, as a **measure of uncertainty** of the learned GMM. We can use MSS and MPO of the learned GMM to set the switching time E_c, batch sizes $\{b_i^{>1}\}_{i=1}^n$ and even the number of underlying clusters M, when it is unknown (Section C in the Appendix).

4.6 Experiments

Datasets, models and baseline algorithms: We evaluate our proposed method on three benchamark datasets, including: MNIST (LeCun et al. 2010), FMNIST (Xiao et al. 2017) and CIFAR10 (Krizhevsky 2009), with heterogeneous data distributions from covariate shift (rotation; $P_i(x)$ varies across clusters) (Kairouz et al. 2021; Werner et al. 2023) and concept shift

(label flip; $P_i(y|x)$ varies across clusters) (Werner et al. 2023), which are the commonly used data splits for clustered FL (see Appendix A). We consider four clusters of clients indexed by $m \in \{0, 1, 2, 3\}$ with $\{3, 6, 6, 6\}$ clients, where the smallest cluster is considered as the minority cluster. We compare our method with most recent related DPFL algorithms under an equal total sample-level privacy budget ϵ: 1. Global (Noble et al. 2021): clients run DPSGD locally and send their model updates to the server for aggregation and learning one global model 2. Local (Liu et al. 2022): clients do not participate FL and learn a local model by running DPSGD on their local data 3. A DP extension of IFCA (Ghosh et al. 2020; Liu et al. 2022): local loss/accuracy-based clustering performed by clients on existing cluster models 4. MR-MTL (Liu et al. 2022): uses model personalization to learn one model for each client 5. O-DPCFL: an oracle algorithm which has the knowledge of the true clusters from the first round. For R-DPCFL and IFCA, *clients run exponential mechanism locally* to privatize their local cluster selections (line 21 of Algorithm 2). In the following, we provide a brief justification for choosing the baseline algorithms above.

4.6.1 Baseline selection

When extending existing personalized and clustered FL algorithms to DPFL settings, we are mostly interested in those with little to no additional local dataset queries to prevent extra noise for DPSGD under a fixed total privacy budget ϵ. For instance, the family of mean-regularized multi-task learning methods (MR-MTL) (Dinh et al. 2020; Evgeniou and Pontil 2004; Hanzely et al. 2020; Hanzely and Richtárik 2021) provide model personalization *without an additional privacy overhead*. Despite this, it is noteworthy that MR-MTL relies on optimal hyperparameter tuning which leads to a potential privacy overhead (Liu and Talwar 2019; Liu et al. 2022; Papernot and Steinke 2022). While resembling MR-MTL, Ditto (Li et al. 2020b) has extra local computations, which makes it a less attractive personalization algorithm. Hence, we adopt MR-MTL (Liu et al. 2022) as a baseline personalized FL algorithm. Similarly, multi-task learning algorithms of Smith et al. 2017 and Marfoq et al. 2021 as well as gradient-based clustered FL algorithm of Sattler et al. 2019 benefit from additional training and training restarts, which lead to high privacy overhead for them, making them less attractive. In contrast, the aforementioned loss-based clustered FL algorithms (Ghosh et al. 2020; Mansour et al. 2020; Ruan and Joe-Wong 2021) can be managed to have a low privacy overhead (see Appendix B.5), and we use it as a DP clustered FL baseline algorithm.

4.6.2 Results

Liu et al. (2022) observed that under sample-level differential privacy and "mil data heterogeneity", federation is more beneficial than local training, because, despite the data heterogeneity across the clients, the model aggregation (averaging) on the server results in reduction of the DP noise in clients' model updates. However, when there is a "high structured data heterogeneity" across clusters of clients, the level of heterogeneity is remarkable. Hence, learning one global model through FL is not beneficial, as one single model can barely adapt to the high level of data

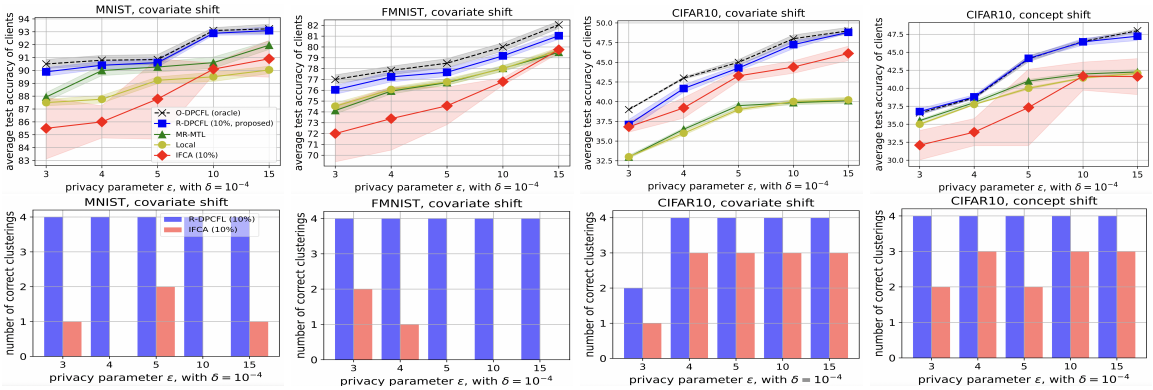


Figure 4.6: **Top:** Average test accuracy for different total privacy budgets ϵ. Results are from four different runs. 10% means performing local clustering by clients only in 10% of the total number of rounds; i.e. rounds $E_c \leq e \leq E_c + \lfloor \frac{E}{10} \rfloor$ for R-DPCFL and rounds $1 \leq e \leq 1 + \lfloor \frac{E}{10} \rfloor$ for IFCA (see Appendix B.5). **Figure D.3 in the Appendix includes the** Global **baseline too**. **Bottom:** Number of times (out of 4 runs) that R-DPCFL and IFCA successfully detect the underlying cluster structure of all existing clients.

heterogeneity across the clusters. Therefore, in DP clustered FL systems, local training and model personalization can be better options than global training, as they diminish the adverse effect of the high data heterogeneity. Furthermore, *if one can detect the underlying clusters*, one can perform FL in each of them separately and simultaneously benefit from 1. eliminating the effect of data heterogeneity across clusters; 2. reduction of the DP noise by running FL aggregation within each cluster. Hence, *if the clustering task is done accurately* (what R-DPCFL is designed for), we can expect a further improvement over local training and model personalization.

RQ1: How does R-DPCFL **perform in practice?** Figure 4.6 shows the average test accuracy across clients for four datasets. As can be observed, R-DPCFL outperforms the baseline algorithms, and this can be attributed to the robust clustering method of R-DPCFL, which additionally benefits from the unused information in clients' model updates in the first round and leads to correct clustering of clients (Figure 4.6, bottom row). While R-DPCFL performs close to the oracle algorithm, IFCA has a lower performance due to its errors in detecting the underlying true clusters. For instance, IFCA has a clearly low clustering accuracy on MNIST and FMNIST, which leads it to perform even worse than Local and MR-MTL. In contrast, it has a better clustering performance on CIFAR10 (covariate and concept shifts) and outperforms the two baselines. On the other hand, the reason behind the low performance of MR-MTL is that it performs personalization on a global model, which in turn has a low quality due to being obtained from federation across "all" clients (hence adversely affected by the high data heterogeneity). Similarly, Local, which performs close to MR-MTL, cannot outperform R-DPCFL, as it does not benefit from the DP noise reduction by FL aggregation within each cluster.

RQ2: How does the minority cluster benefit from R-DPCFL**?** Figure 4.7 compares different algorithms based on the average test accuracy of the clients belonging to the minority cluster. R-DPCFL leads to a better overall performance for the minority clients, by virtue of its correct and robust cluster detection. Correct detection of the minority cluster prevents it from getting mixed

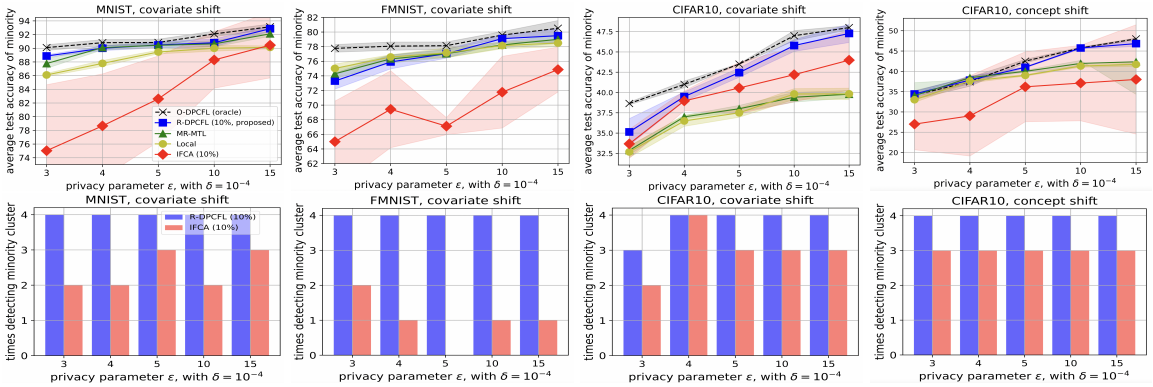


Figure 4.7: **Top:** Average test accuracy across clients belonging to the minority cluster for different total privacy budgets ϵ, averaged over four different runs (random seeds). **Bottom:** Number of times (out of 4 runs) that R-DPCFL and IFCA successfully detect the minority cluster.

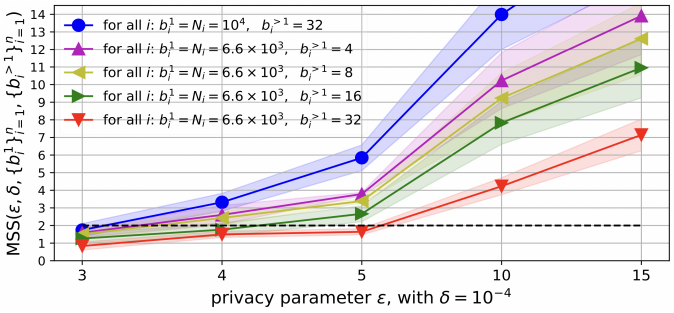


Figure 4.8: MSS score v.s. ϵ for two different local dataset sizes. A small local dataset size can be compensated for by using smaller batch sizes $\{b_i^{>1}\}_{i=1}^n$ to get a larger MSS score.

with other majority clusters and leads to a utility improvement for its clients. In contrast, IFCA has a lower success rate in detecting the minority cluster (Figure 4.7, bottom row) and provides a lower overall performance for them. Similarly, Local and MR-MTL lead to a low performance for the minority, as they are conditioned on a global model that is learned from federation across all clients and provides a low performance for the minority. Detecting and improving the performance of minority clusters is important, as the different clusters in DP clustered FL systems may not be of the same size, and failure in this task leads to a low performance for the ones with smaller sizes.

RQ3: What if clients have small local datasets? While we envision the proposed approach being more applicable to cross-silo FL, where datasets are large, it is still worth exploring how beneficial it can be under scarce local data. In the previous sections, we analyzed the benefits of using a full batch size ($b_i^1 = N_i$) in the first round and found that it leads to a GMM with a higher MSS confidence score. The score of the learned GMM can strongly predict whether the underlying true clusters will be detected: an MSS above 2 almost always yields to correct detection of the underlying clusters (see Figure D.5 in the Appendix for experimental results). On the other hand the MSS score depends on four sets of parameters: $\epsilon, \delta, \{b_i^1\}_{i=1}^n$, and $\{b_i^{>1}\}_{i=1}^n$. For fixed (ϵ, δ), larger $\{b_i^1\}_{i=1}^n$ and smaller $\{b_i^{>1}\}_{i=1}^n$ increase MSS (Lemma 4). When we ue full batch sizes in the

first round, we have $b_i^1 = N_i$ (for all i). Hence, smaller local datasets result in lower confidence in the learned GMM. Nevertheless, this can be compensated for by using even smaller $\{b_i^{>1}\}_{i=1}^n$. Figure 4.8 compares two different dataset sizes under varying ϵ. As observed, for smaller local dataset sizes, reducing $\{b_i^{>1}\}_{i=1}^n$ can help obtain less noisy model updates $\{\Delta\tilde{\boldsymbol{\theta}}i^1\}_{i=1}^n$, improve the MSS score of the learned GMM and consequently, enable a successful client clustering.

4.7 Summary

We proposed a DP clustered FL algorithm, which addresses sample-level privacy in FL systems with structured data heterogeneity. By clustering clients based on both their model updates and training loss/accuracy values, and mitigating noise impacts with large initial batch sizes, our approach enhances clustering accuracy and mitigates DP's disparate impact on utility, all with minimal computational overhead. Moreover, the robustness to noise and easy parameter selection of the proposed approach shows its applicability to DP clustered FL settings. While envisioned for DPFL systems with large local datasets, the method is capable of compensating for moderate dataset sizes by using smaller batch sizes after the first round. In the future, we aim to extend this approach to be suitable for scarce data scenarios, which might be found in cross-device FL.

Chapter 5

Conclusion

Despite the fast progress in FL and the proliferation of many new algorithms, its wide application needs understanding its limitations and finding remedies for them. This thesis examines three aspects of FL, including 1. convergence analysis and acceleration of FL algorithms; 2. utility improvement in private FL systems under heterogeneous privacy preferences; and 3. performance fairness in private FL systems under structured data heterogeneity.

In Chapter 2, we analyze the convergence of some widely used FL algorithms and show that many of them can be unified into a grand scheme obtained from an operator splitting point of view. This unification lets us refine their convergence analysis, accelerate their convergence and propose new algorithms. This unification and acceleration diminishes the communication overhead of the existing FL algorithms.

Chapter 3 investigates differentially private FL systems with heterogeneous privacy preferences for different clients. We show that a naive aggregation strategy for the server in these systems leads to a low utility. Therefore, we propose a noise-aware aggregation strategy for the server to assign aggregation weights to clients based on the noise level in their uploaded model updates at the end of each round. This strategy leads to a reduction of the noise in the aggregated model updates and utility improvement for the system. Besides, the strategy provides robustness to malicious clients who may deliberately send falsified information about their privacy preferences to the server to attack the system's utility.

Finally, in Chapter 4, we propose a robust algorithm for differentially private FL systems with structured data heterogeneity across clients. In such systems, correct clustering of clients based on their data similarity is vital for achieving high utility. However, the existing clustered FL algorithms are prone to mistakes in DPFL systems, due to the existing DP noise. Our proposed method achieves high clustering accuracy by clustering clients based on both their model updates in the first few rounds and their loss values in the subsequent rounds. Besides, it leverages using a large batch size in the first round to reduce the noise in clients' model updates as much as possible. The proposed clustering strategy achieves high accuracy in clustering clients, which consequently leads to an improvement in the system's overall performance as well as that of the minority groups of clients.

References

Abadi, M., A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang (2016). "Deep Learning with Differential Privacy". In: *ACM SIGSAC Conference on Computer and Communications Security*.

Alaggan, M., S. Gambs, and A.-M. Kermarrec (2017). "Heterogeneous Differential Privacy". *Journal of Privacy and Confidentiality*.

Aleyner, A. and S. Reich (2009). "Random Products of Quasi-Nonexpansive Mappings in Hilbert Space". *Journal of Convex Analysis*.

Anderson, D. G. (1965). "Iterative Procedures for Nonlinear Integral Equations". *Journal of ACM*.

Arthur, D. and S. Vassilvitskii (2007). "k-means++: the advantages of careful seeding". In: *ACM-SIAM Symposium on Discrete Algorithms*.

Augenstein, S., H. B. McMahan, D. Ramage, S. Ramaswamy, P. Kairouz, M. Chen, R. Mathews, and B. A. y Arcas (2020). "Generative Models for Effective ML on Private, Decentralized Datasets". In: *International Conference on Learning Representations (ICLR)*.

Auslender, A. (1969). "Méthodes numériques pour la résolution des problèmes d'optimisation avec contraintes".

Bagdasaryan, E. and V. Shmatikov (2019). "Differential Privacy Has Disparate Impact on Model Accuracy". In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Bagdasaryan, E., A. Veit, Y. Hua, D. Estrin, and V. Shmatikov (2020). "How To Backdoor Federated Learning". In: *International Conference on Artificial intelligence and statistics (AISTATS)*.

Bauschke, H. H. and S. G. Kruk (2004). "Reflection-Projection Method for Convex Feasibility Problems with an Obtuse Cone". *Journal of Optimization Theory and Applications*.

Bauschke, H. H. and P. L. Combettes (2017). "Convex Analysis and Monotone Operator Theory in Hilbert Spaces". Springer.

Bauschke, H. H., P. L. Combettes, and S. Reich (2005). "The asymptotic behavior of the composition of two resolvents". *Nonlinear Analysis: Theory, Methods & Applications*.

Bertsimas, D., V. F. Farias, and N. Trichakis (2011). "The price of fairness". *Operations research*.

Bhagoji, A. N., S. Chakraborty, P. Mittal, and S. Calo (2019). "Analyzing Federated Learning through an Adversarial Lens". In: *International Conference on Machine Learning (ICML)*.

Biernacki, C., G. Celeux, and G. Govaert (2003). "Choosing starting values for the EM algorithm for getting the highest likelihood in multivariate Gaussian mixture models". *Computational, Statistics and Data Analysis*.

Billingsley, P. (1995). "Probability and Measure". JohnWileySons.

Boenisch, F., C. Mühl, A. Dziedzic, R. Rinberg, and N. Papernot (2023). "Have it your way: Individualized Privacy Assignment for DP-SGD". In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Brézis, H. and F. E. Browder (1976). "Nonlinear ergodic theorems". *Bulletin of the American Mathematical Society*.

Briggs, C., Z. Fan, and P. András (2020). "Federated learning with hierarchical clustering of local updates to improve training on non-IID data". *International Joint Conference on Neural Networks (IJCNN)*.

Bruck, R. E. (1977). "On the weak convergence of an ergodic iteration for the solution of variational inequalities for monotone operators in Hilbert space". *Journal of Mathematical Analysis and Applications*.

Bun, M. and T. Steinke (2016). "Concentrated Differential Privacy: Simplifications, Extensions, and Lower Bounds". In: *Theory of Cryptography*. Springer Berlin Heidelberg.

Caldas, S., P. Wu, T. Li, J. Konecny, H. B. McMahan, V. Smith, and A. Talwalkar (2019). "LEAF: A benchmark for federated settings". In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Candès, E. J., X. Li, Y. Ma, and J. Wright (2009). "Robust principal component analysis?" *Journal of ACM*.

Canonne, C. L., G. Kamath, and T. Steinke (2020). "The Discrete Gaussian for Differential Privacy". In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Charles, Z. and J. Konečný (2021). "Convergence and Accuracy Trade-Offs in Federated Learning and Meta-Learning". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Chathoth, A. K., C. P. Necciai, A. Jagannatha, and S. Lee (2022). "Differentially Private Federated Continual Learning with Heterogeneous Cohort Privacy". In: *IEEE International Conference on Big Data (Big Data)*.

Cimmino, G. (1938). "Calcolo Approssimato Per le Soluzioni dei Sistemi di Equazioni Lineari". *La Ricerca Scientifica, Series II*.

Cummings, R., V. Gupta, D. Kimpara, and J. Morgenstern (2019). "On the Compatibility of Privacy and Fairness". In: *Conference on User Modeling, Adaptation and Personalization*.

Deng, Y., M. M. Kamani, and M. Mahdavi (2020). "Adaptive Personalized Federated Learning". *ArXiv*.

Diao, E., J. Ding, and V. Tarokh (2021). "HeteroFL: Computation and Communication Efficient Federated Learning for Heterogeneous Clients". In: *International Conference on Learning Representations (ICLR)*.

Dinh, C. T., N. Tran, and J. Nguyen (2020). "Personalized Federated Learning with Moreau Envelopes". In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Donahue, K. and J. Kleinberg (2021). "Model-sharing Games: Analyzing Federated Learning Under Voluntary Participation". *AAAI Conference on Artificial Intelligence*.

Douglas Jr., J. and H. H. Rachford Jr. (1956). "On the Numerical Solution of Heat Conduction Problems in Two and Three Space Variables". *Transactions of the American Mathematical Society*.

Duchi, J. C., M. I. Jordan, and M. J. Wainwright (2013). "Local privacy and statistical minimax rates". In: *Annual Allerton Conference on Communication, Control, and Computing (Allerton)*.

Duchi, J. C., M. J. Wainwright, and M. I. Jordan (2018). "Minimax Optimal Procedures for Locally Private Estimation". *Journal of the American Statistical Association*.

Dwork, C. (2011). "A firm foundation for private data analysis". *Communications of the ACM*.

Dwork, C., M. Hardt, T. Pitassi, O. Reingold, and R. Zemel (2012). "Fairness through awareness". In: *Theoretical Computer Science Conference*. Association for Computing Machinery.

Dwork, C., K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor (2006a). "Our Data, Ourselves: Privacy via Distributed Noise Generation". In: *International Conference on The Theory and Applications of Cryptographic Techniques*.

Dwork, C., F. McSherry, K. Nissim, and A. Smith (2006b). "Calibrating Noise to Sensitivity in Private Data Analysis". In: *Conference on Theory of Cryptography*. Springer-Verlag.

Dwork, C. and A. Roth (2014). "The Algorithmic Foundations of Differential Privacy". *Foundations and Trends® in Theoretical Computer Science*.

Dwork, C., G. N. Rothblum, and S. P. Vadhan (2010). "Boosting and Differential Privacy". *IEEE Annual Symposium on Foundations of Computer Science*.

Eckstein, J. and D. P. Bertsekas (1992). "On the Douglas—Rachford splitting method and the proximal point algorithm for maximal monotone operators". *Mathematical Programming*.

Esipova, M. S., A. A. Ghomi, Y. Luo, and J. C. Cresswell (2022). "Disparate Impact in Differential Privacy from Gradient Misalignment". In: *International Conference on Learning Representations (ICLR)*.

Evgeniou, T. and M. Pontil (2004). "Regularized multi–task learning". In: *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Fallah, A., A. Makhdoumi, A. Malekian, and A. Ozdaglar (2023). "Optimal and differentially private data acquisition: Central and local mechanisms". *Operations Research*.

Farrand, T., F. Mireshghallah, S. Singh, and A. Trask (2020). "Neither Private Nor Fair: Impact of Data Imbalance on Utility and Fairness in Differential Privacy". In: *Proceedings of the 2020 Workshop on Privacy-Preserving Machine Learning in Practice*.

Fioretto, F., C. Tran, P. V. Hentenryck, and K. Zhu (2022). "Differential Privacy and Fairness in Decisions and Learning Tasks: A Survey". In: *International Joint Conference on Artificial Intelligence*.

Fu, A., J. Zhang, and S. Boyd (2020). "Anderson Accelerated Douglas–Rachford Splitting". *SIAM Journal on Scientific Computing*.

Gabay, D. (1983). "Applications of the Method of Multipliers to Variational Inequalities". Studies in Mathematics and Its Applications.

Geiping, J., H. Bauermeister, H. Dröge, and M. Moeller (2020). "Inverting Gradients - How easy is it to break privacy in federated learning?" In: *Advances in Neural Informatioon Processing Systems (NeurIPS)*.

Geyer, R. C., T. Klein, and M. Nabi (2017). "Differentially Private Federated Learning: A Client Level Perspective". In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Ghadimi, S. and G. Lan (2013). "Stochastic First- and Zeroth-Order Methods for Nonconvex Stochastic Programming". *SIAM Journal on Optimization*.

Ghosh, A., J. Chung, D. Yin, and K. Ramchandran (2020). "An Efficient Framework for Clustered Federated Learning". In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Girgis, A. M., D. Data, S. N. Diggavi, P. Kairouz, and A. T. Suresh (2021). "Shuffled Model of Differential Privacy in Federated Learning". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Gorbunov, E., F. Hanzely, and P. Richtárik (2021). "Local SGD: Unified theory and new efficient methods". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Gur-Ari, G., D. A. Roberts, and E. Dyer (2018). "Gradient Descent Happens in a Tiny Subspace". *ArXiv*.

Hanzely, F., S. Hanzely, S. Horváth, and P. Richtárik (2020). "Lower Bounds and Optimal Algorithms for Personalized Federated Learning". In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Hanzely, F. and P. Richtárik (2021). "Federated Learning of a Mixture of Global and Local Models". ArXiv.

He, C. et al. (2020). "FedML: A research library and benchmark for federated machine learning". ArXiv.

He, K., X. Zhang, S. Ren, and J. Sun (2015). "Deep Residual Learning for Image Recognition". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Heo, G., J. Seo, and S. E. Whang (2023). "Personalized DP-SGD using Sampling Mechanisms". *ArXiv*.

Hitaj, B., G. Ateniese, and F. Pérez-Cruz (2017). "Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning". *ACM SIGSAC Conference on Computer and Communications Security*.

Hu, Z., K. Shaloudegi, G. Zhang, and Y. Yu (2022). "Federated Learning Meets Multi-Objective Optimization". *IEEE Transactions on Network Science and Engineering*.

Huang, W., S. Zhou, T. Zhu, Y. Liao, C. Wu, and S. Qiu (2020). "Improving Laplace Mechanism of Differential Privacy by Personalized Sampling". In: *IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*.

Huo, Z., Q. Yang, B. Gu, L. Carin, and H. Huang (2020). "Faster On-Device Training Using New Federated Momentum Algorithm". ArXiv.

Jorgensen, Z., T. Yu, and G. Cormode (2015). "Conservative or liberal? Personalized differential privacy". In: *IEEE 31st International Conference on Data Engineering (ICDE)*.

Kairouz, P. et al. (2021). "Advances and Open Problems in Federated Learning". ArXiv: 1912.04977.

Kang, J., R. Pedarsani, and K. Ramchandran (2023). "The Fair Value of Data Under Heterogeneous Privacy Constraints". *Transactions on Machine Learning Research (TMLR)*.

Karimireddy, S. P., W. Guo, and M. I. Jordan (2022). "Mechanisms that Incentivize Data Sharing in Federated Learning". *ArXiv*.

Khaled, A., K. Mishchenko, and P. Richtarik (2020a). "Tighter Theory for Local SGD on Identical and Heterogeneous Data". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Khaled, A., K. Mishchenko, and P. Richtárik (2020b). "First Analysis of Local GD on Heterogeneous Data". ArXiv:1909.04715.

Kotsogiannis, I., S. Doudalis, S. Haney, A. Machanavajjhala, and S. Mehrotra (2020). "One-sided Differential Privacy". In: *IEEE 36th International Conference on Data Engineering (ICDE)*.

Krizhevsky, A. (2009). "Learning Multiple Layers of Features from Tiny Images".

Lan, T., D. Kao, M. Chiang, and A. Sabharwal (2010). "An axiomatic theory of fairness in network resource allocation". In: *IEEE INFOCOM*.

LeCun, Y., C. Cortes, and C. Burges (2010). "MNIST handwritten digit database".

Li, D. and J. Wang (2019). "FedMD: Heterogenous Federated Learning via Model Distillation". *ArXiv*.

Li, T., A. Beirami, M. Sanjabi, and V. Smith (2020a). "Tilted Empirical Risk Minimization". In: *International Conference on Learning Representations (ICLR)*.

Li, T., S. Hu, A. Beirami, and V. Smith (2020b). "Ditto: Fair and Robust Federated Learning Through Personalization". In: *International Conference on Machine Learning (ICML)*.

Li, T., A. K. Sahu, A. Talwalkar, and V. Smith (2019). "Federated learning: Challenges, methods, and future directions". ArXiv:1908.07873.

Li, T., A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith (2020c). "Federated Optimization in Heterogeneous Networks". In: *Machine Learning and Systems*.

Li, T., M. Sanjabi, A. Beirami, and V. Smith (2020d). "Fair Resource Allocation in Federated Learning". In: *International Conference on Learning Representations (ICLR)*.

Li, X., K. Huang, W. Yang, S. Wang, and Z. Zhang (2020e). "On the Convergence of FedAvg on Non-IID Data". In: *International Conference on Learning Representations (ICLR)*.

Lions, J. L. and R. Temam (1966). "Une méthode d'éclatement pes opérateurs et des contraintes en calcul des variations". *Comptes rendus mathématiques de l'Académie des Sciences, Paris*.

Lions, P.-L. (1978). "Une methode iterative de resolution d'une inequation variationnelle". *Israel Journal of Mathematics*.

Lions, P.-L. and B. Mercier (1979). "Splitting Algorithms for the Sum of Two Nonlinear Operators". *SIAM Journal on Numerical Analysis*.

Liu, J. and K. Talwar (2019). "Private selection from private candidates". In: *Annual ACM SIGACT Symposium on Theory of Computing*.

Liu, J., J. Lou, L. Xiong, J. Liu, and X. Meng (2021a). "Projected Federated Averaging with Heterogeneous Differential Privacy". *VLDB Endowment.*

Liu, R., Y. Cao, H. Chen, R. Guo, and M. Yoshikawa (2021b). "FLAME: Differentially Private Federated Learning in the Shuffle Model". In: *AAAI Conference on Artificial Intelligence*.

Liu, Y., Y. Kang, C. Xing, T. Chen, and Q. Yang (2020). "A Secure Federated Transfer Learning Framework". *IEEE Intelligent Systems*.

Liu, Z., S. Hu, Z. S. Wu, and V. Smith (2022). "On Privacy and Personalization in Cross-Silo Federated Learning". In: *Advances ibn Neural Information Processing Systems (NeurIPS)*.

Ma, J., L. Xu, and M. I. Jordan (2000). "Asymptotic Convergence Rate of the EM Algorithm for Gaussian Mixtures". *Neural Computation*.

Malekmohammadi, S., K. Shaloudegi, Z. Hu, and Y. Yu (2021a). "An Operator Splitting View of Federated Learning". *ArXiv*.

— (2021b). "Splitting Algorithms for Federated Learning". In: *Machine Learning and Principles and Practice of Knowledge Discovery in Databases*.

— (2023). "A Unifying Framework for Federated Learning". In: *Federated and Transfer Learning*. Springer.

Malekmohammadi, S., A. Taik, and G. Farnadi (2025). "Differentially Private Clustered Federated Learning".

Malekmohammadi, S. and Y. Yu (2025). "Semi-Variance Reduction for Fair Federated Learning". arXiv: 2406.16193 [cs.LG].

Malekmohammadi, S., Y. Yu, and Y. Cao (2024). "Noise-Aware Algorithm for Heterogeneous Differentially Private Federated Learning". In: *International Conference on Machine Learning (ICML)*.

Malinovskiy, G., D. Kovalev, E. Gasanov, L. Condat, and P. Richtarik (2020). "From Local SGD to Local Fixed-Point Methods for Federated Learning". In: *International Conference on Machine Learning (ICML)*.

Mansour, Y., M. Mohri, J. Ro, and A. T. Suresh (2020). "Three Approaches for Personalization with Applications to Federated Learning". arXiv.

Marfoq, O., G. Neglia, A. Bellet, L. Kameni, and R. Vidal (2021). "Federated Multi-Task Learning under a Mixture of Distributions". In: *Neural Information Processing Systems (NeurIPS))*.

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

McMahan, B., E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas (2017a). "Communication-efficient learning of deep networks from decentralized data". In: *Artificial intelligence and statistics (AISTATS)*.

McMahan, B., E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas (2017b). "Communication-Efficient Learning of Deep Networks from Decentralized Data". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

McMahan, H. B., G. Andrew, U. Erlingsson, S. Chien, I. Mironov, N. Papernot, and P. Kairouz (2019). "A General Approach to Adding Differential Privacy to Iterative Training Procedures". ArXiv.

McMahan, H. B., D. Ramage, K. Talwar, and L. Zhang (2018). "Learning Differentially Private Recurrent Language Models". In: *International Conference on Learning Representations (ICLR)*.

Mironov, I. (2017). "Rényi Differential Privacy". In: *IEEE Computer Security Foundations Symposium (CSF)*.

Mironov, I., K. Talwar, and L. Zhang (2019). "Rényi Differential Privacy of the Sampled Gaussian Mechanism". *CoRR*.

Mohri, M., G. Sivek, and A. T. Suresh (2019). "Agnostic federated learning". In: *International Conference on Machine Learning (ICML)*.

Nasr, M., R. Shokri, and A. Houmansadr (2019). "Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning". In: *IEEE Symposium on Security and Privacy (SP)*.

Niu, B., Y. Chen, B. Wang, J. Cao, and F. Li (2020). "Utility-aware Exponential Mechanism for Personalized Differential Privacy". In: *IEEE Wireless Communications and Networking Conference (WCNC)*.

Noble, M., A. Bellet, and A. Dieuleveut (2021). "Differentially Private Federated Learning on Heterogeneous Data". In: *International Conference on Artificial intelligence and statistics (AISTATS)*.

Papernot, N. and T. Steinke (2022). "Hyperparameter Tuning with Renyi Differential Privacy".

Passty, G. B. (1979). "Ergodic convergence to a zero of the sum of monotone operators in Hilbert space". *Journal of Mathematical Analysis and Applications*.

Pathak, R. and M. J. Wainwright (2020). "FedSplit: An algorithmic framework for fast federated optimization". In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Peaceman, D. W. and H. H. Rachford Jr. (1955). "The Numerical Solution of Parabolic and Elliptic Differential Equations". *Journal of the Society for Industrial and Applied Mathematics*.

Pentyala, S., N. Neophytou, A. Nascimento, M. D. Cock, and G. Farnadi (2022). "PrivFairFL: Privacy-Preserving Group Fairness in Federated Learning". ArXiv.

Reddi, S., Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečny, S. Kumar, and H. B. McMahan (2020). "Adaptive Federated Optimization". ArXiv.

Reisizadeh, A., F. Farnia, R. Pedarsani, and A. Jadbabaie (2020). "Robust Federated Learning: The Case of Affine Distribution Shifts". In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Rigaki, M. and S. García (2020). "A Survey of Privacy Attacks in Machine Learning". *ArXiv*.

Rockafellar, R. T. (2019). "Progressive Decoupling of Linkages in Optimization and Variational Inequalities with Elicitable Convexity or Monotonicity". *Set-Valued and Variational Analysis*.

Rockafellar, R. T. and R. J.-B. Wets (1998). "Variational Analysis". Springer.

Rogers, R. and T. Steinke (2021). "A better privacy analysis of the exponential mechanism".

Ruan, Y. and C. Joe-Wong (2021). "FedSoft: Soft Clustered Federated Learning with Proximal Local Updating". *CoRR*.

Sattler, F., K.-R. Müller, and W. Samek (2019). "Clustered Federated Learning: Model-Agnostic Distributed Multitask Optimization Under Privacy Constraints". *IEEE Transactions on Neural Networks and Learning Systems*.

Shi, W., A. Cui, E. Li, R. Jia, and Z. Yu (2021). "Selective Differential Privacy for Language Modeling". *ArXiv*.

Smith, V., C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar (2017). "Federated Multi-Task Learning". In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Spingarn, J. E. (1983). "Partial inverse of a monotone operator". *Applied Mathematics and Optimization*.

Spingarn, J. E. (1985). "Applications of the method of partial inverses to convex programming: Decomposition". *Mathematical Programming*.

Sun, Z., P. Kairouz, A. T. Suresh, and H. B. McMahan (2019). "Can You Really Backdoor Federated Learning?" ArXiv:1911.07963.

Themelis, A. and P. Patrinos (2020). "Douglas–Rachford Splitting and ADMM for Nonconvex Optimization: Tight Convergence Results". *SIAM Journal on Optimization*.

Tran, C., F. Fioretto, and P. V. Hentenryck (2020). "Differentially Private and Fair Deep Learning: A Lagrangian Dual Approach". *ArXiv*.

Tseng, P. (1992). "On the Convergence of the Products of Firmly Nonexpansive Mappings". *SIAM Journal on Optimization*.

— (2000). "A Modified Forward-Backward Splitting Method for Maximal Monotone Mappings". *SIAM Journal on Control and Optimization*.

Wang, H., M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni (2020). "Federated Learning with Matched Averaging". In: *International Conference on Learning Representations (ICLR)*.

Wang, Z., M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi (2019). "Beyond Inferring Class Representatives: User-Level Privacy Leakage From Federated Learning". *IEEE INFOCOM*.

Werner, M., L. He, M. Jordan, M. Jaggi, and S. P. Karimireddy (2023). "Provably Personalized and Robust Federated Learning". *Transactions on Machine Learning Research (TMLR)*.

Wu, Y., S. Zhang, W. Yu, Y. Liu, Q. Gu, D. Zhou, H. Chen, and W. Cheng (2023). "Personalized Federated Learning under Mixture of Distributions". In: *International Conference on Machine Learning (ICML)*.

Xiao, H., K. Rasul, and R. Vollgraf (2017). "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms". *CoRR*.

Xu, D., W. Du, and X. Wu (2021). "Removing Disparate Impact on Model Accuracy in Differentially Private Stochastic Gradient Descent". *ACM SIGKDD Conference on Knowledge Discovery & Data Mining*.

Yang, Q., Y. Liu, T. Chen, and Y. Tong (2019). "Federated Machine Learning: Concept and Applications". *ACM Transactions on Intelligent Systems and Technology*.

Yu, D., G. Kamath, J. Kulkarni, T.-Y. Liu, J. Yin, and H. Zhang (2023). "Individual Privacy Accounting for Differentially Private Stochastic Gradient Descent". *Transactions on Machine Learning Research (TMLR)*.

Yu, L., L. Liu, C. Pu, M. E. Gursoy, and S. Truex (2019). "Differentially Private Model Publishing for Deep Learning". *IEEE Symposium on Security and Privacy (SP)*.

Yu, Y. (2013). "Better Approximation and Faster Algorithm Using the Proximal Average". In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Yu, Y., X. Zheng, M. Marchetti-Bowick, and E. P. Xing (2015). "Minimizing Nonconvex Non-Separable Functions". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Yurochkin, M., M. Agarwal, S. Ghosh, K. Greenewald, N. Hoang, and Y. Khazaeni (2019). "Bayesian Nonparametric Federated Learning of Neural Networks". In: *International Conference on Machine Learning (ICML)*.

Zhang, G., S. Malekmohammadi, X. Chen, and Y. Yu (2023). "Proportional Fairness in Federated Learning". *Transactions on Machine Learning Research (TMLR)*.

Zhang, J., B. O'Donoghue, and S. Boyd (2020). "Globally Convergent Type-I Anderson Acceleration for Nonsmooth Fixed-Point Iterations". *SIAM Journal on Optimization*.

Zhang, M., K. Sapra, S. Fidler, S. Yeung, and J. M. Alvarez (2021). "Personalized Federated Learning with First Order Model Optimization". In: *International Conference on Learning Representations (ICLR)*.

Zhao, Y., J. Zhao, M. Yang, T. Wang, N. Wang, L. Lyu, D. Niyato, and K.-Y. Lam (2021). "Local Differential Privacy-Based Federated Learning for Internet of Things". *IEEE Internet of Things Journal*.

Zhou, J., Z. Su, J. Ni, Y. Wang, Y. Pan, and R. Xing (2022). "Personalized Privacy-Preserving Federated Learning: Optimized Trade-off Between Utility and Privacy". In: *IEEE Global Communications Conference (GLOBECOM)*.

Zhou, Y., S. Wu, and A. Banerjee (2021). "Bypassing the Ambient Dimension: Private SGD with Gradient Subspace Identification". In: *International Conference on Learning Representations (ICLR)*.

Zhu, L., Z. Liu, and S. Han (2019). "Deep Leakage from Gradients". In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Appendix A

Appendix of Chapter 2

A Experimental Setup

In this section we provide more experimental details that are deferred from the main paper.

A.1 Experimental setup: least squares and logistic regression

For simulating an instance of the aforementioned least squares and logistic regression problems, we follow the experimental setup of Pathak and Wainwright (2020).

Least squares regression: We consider a set of n clients with local loss functions $f_i(\boldsymbol{\theta}) := \frac{1}{2}\|A_i\boldsymbol{\theta} - \mathbf{b}_i\|_2^2$, and the main goal is to solve the following minimization problem:

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^p} \; f(\boldsymbol{\theta}) := \sum_{i=1}^{n} f_i(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{n} \|A_i\boldsymbol{\theta} - \mathbf{b}_i\|_2^2,$$

where $\boldsymbol{\theta} \in \mathbb{R}^p$ is the optimization variable. For each client i, the response vector $\mathbf{b}_i \in \mathbb{R}^{N_i}$ is related to the design matrix $A_i \in \mathbb{R}^{N_i\times d}$ via the linear model

$$\mathbf{b}_i = A_i\boldsymbol{\theta}^\star + \boldsymbol{\epsilon}_i,$$

where $\boldsymbol{\epsilon}_i \sim N(0, \sigma^2 I_{N_i})$ for some $\sigma > 0$ is the noise vector. The design matrix $A_i \in \mathbb{R}^{N_i\times d}$ is generated by sampling its elements from a standard normal, $A_i^{k,l} \sim N(0,1)$. We instantiated the problem with the following set of parameters:

$$n = 25, \;\; d = 100, \;\; N_i = 5000, \;\; \sigma^2 = 0.25.$$

Binary logistic regression: There are n clients and the design matrices $A_i \in \mathbb{R}^{N_i \times d}$ for $i = 1, \ldots, n$ are generated as described before. Each client i has a label vector $\mathbf{b}_i \in \{-1, 1\}^{N_i}$. The conditional probability of observing $\mathbf{b}_{ij} = 1$ (the j-th label in $\mathbf{b}_i$) is

$$\mathbf{P}\{\mathbf{b}_{ij} = 1\} = \frac{e^{\mathbf{a}_{ij}^\top \boldsymbol{\theta}_0}}{1 + e^{\mathbf{a}_{ij}^\top \boldsymbol{\theta}_0}}, \quad j = 1, \ldots, N_i,$$

where $\mathbf{a}_{ij}$ is the j-th row of A_i. Also, $\boldsymbol{\theta}_0 \in \mathbb{R}^p$ is fixed and sampled from $N(0, 1)$. Having generated the design matrices A_i and sampled the labels $\mathbf{b}_i$ for all clients, we find the maximum likelihood estimate of $\boldsymbol{\theta}_0$ by solving the following convex program, which has a solution $\boldsymbol{\theta}^*$:

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^p} \; f(\boldsymbol{\theta}) := \sum_{i=1}^{n} f_i(\boldsymbol{\theta}) = \sum_{i=1}^{n} \sum_{j=1}^{N_i} \log(1 + e^{-b_{ij} \mathbf{a}_{ij}^\top \boldsymbol{\theta}}) + \frac{\|\boldsymbol{\theta}\|_2^2}{2nN_i}.$$

Following Pathak and Wainwright (2020), we set

$$n = 10, \quad d = 100, \quad N_i = 1000.$$

Other parameters: Throughout the experiments, we use local learning rate $\eta = 10^{-5}$ for least squares, and $\eta = 10^{-2}$ for logistic regression, and run FedAvg with $k = 5$ for both least squares and logistic regression, unless otherwise specified. In Figure 2.8, FedAvg is run with $k = 2$.

A.2 Experimental setup: MNIST datasets

We consider a distributed setting with 20 clients. In order to create a non-i.i.d. dataset, we follow a similar procedure as in McMahan et al. (2017b): first we split the data from each class into several shards. Then, each client is randomly assigned a number of shards of data. For example, in Figure 2.7 to guarantee that no client receives data from more than 6 classes, we split each class of MNIST into 12 shards (i.e. a total of 120 shards for the whole dataset), and each client is randomly assigned 6 shards of data. By considering 20 clients, this procedure guarantees that no client receives data from more than 6 classes and the data distribution of each client is different from each other. The local datasets are balanced–all clients have the same amount of training samples. The local data is split into train, validation, and test sets with percentage of 80%, 10%, and 10%, respectively. In this way, each client has 2400 data points for training, 300 for test, and 300 for validation. We use a simple 2-layer CNN model with ReLU activation, the detail of which can be found in Table A.1. To update the local models at each client using its local data, unless otherwise is stated, we apply gradient descent with $\eta = 0.01$ for FedAvg , and gradient descent with $k = 100$ and $\eta = 0.01$ for proximal update in the splitting algorithms.

A.3 Experimental setup: CIFAR-10 dataset

We consider a distributed setting with 10 clients, and use a sub-sampled CIFAR-10 dataset, which is 20% of the training set. In order to create a non-i.i.d. dataset, we follow a similar procedure

Table A.1: MNIST model

Layer	Output Shape	# of Trainable Parameters	Activation	Hyper-parameters
Input	$(1, 28, 28)$	0		
Conv2d	$(10, 24, 24)$	260	ReLU	kernel size =5; strides=$(1, 1)$
MaxPool2d	$(10, 12, 12)$	0		pool size=$(2, 2)$
Conv2d	$(20, 8, 8)$	5,020	ReLU	kernel size =5; strides=$(1, 1)$
MaxPool2d	$(20, 4, 4)$	0		pool size=$(2, 2)$
Flatten	320	0		
Dense	20	6,420	ReLU	
Dense	10	210	softmax	
Total		11,910		

as in McMahan et al. (2017b): first we sort all data points according to their classes. Then, they are split into 100 shards, and each client is randomly assigned 10 shards of data. The local data is split into train, validation, and test sets with percentage of 80%, 10%, and 10%, respectively. In this way, each client has 800 data points for training, 100 for test, and 100 for validation. We use a 2-layer CNN model on the dataset, the details of which can be found in Table A.2. To update the local models at each client using its local data, we apply stochastic gradient descent (SGD) with local batch size $B = 20$ and local learning rate $\eta = 0.1$. All clients participate in each communication round.

Table A.2: CIFAR-10 model

Layer	Output Shape	# of Trainable Params	Activation	Hyper-parameters
Input	$(3, 32, 32)$	0		
Conv2d	$(64, 28, 28)$	4,864	ReLU	kernel =5; strides=$(1, 1)$
MaxPool2d	$(64, 14, 14)$	0		pool size=$(2, 2)$
LocalResponseNorm	$(64, 14, 14)$	0		size=2
Conv2d	$(64, 10, 10)$	102,464	ReLU	kernel =5; strides=$(1, 1)$
LocalResponseNorm	$(64, 10, 10)$	0		size=2
MaxPool2d	$(64, 5, 5)$	0		pool size=$(2, 2)$
Flatten	1,600	0		
Dense	384	614,784	ReLU	
Dense	192	73,920	ReLU	
Dense	10	1,930	softmax	
Total		797,962		

B Additional experimental results

In this section we provide more experimental results that are deferred from the main paper.

B.1 Effect of client sampling

The grand scheme introduced in (2.15)-(2.17) assumes full client participation; however, this might not be always the case. So, we modify the grand scheme to investigate the effect of client participation. The modified version, which is based on the centralized implementation of `FL` algorithms (explained in Section 2.5), is given in Algorithm 3. The simulation results for least squares and logistic regression are shown in Figure B.1. From the results it can be observed that, with partial client participation, none of the aforementioned algorithms converge to the correct solution, especially FedAvg with $k = 1$. Also, FedSplit becomes unstable when client participation is low ($p = 0.5$), while other algorithms e.g., FedPi remain stable.

Algorithm 3: Federated learning with client sampling.

Input: p (user participation probability), $\alpha_t, \beta_t, \gamma_t, \eta_t$
Output: $\bar{\boldsymbol{\Theta}}$ (final global model)

initialize $\mathbf{u}_0$ and $\mathbf{z}_0$ for all clients
Server *(p)*:
  for *each round* $t = 1, 2, \ldots$ **do**
    $S_t \leftarrow$ sample the set of present clients, where each client participates with probability p
    for *each client* i ***in parallel*** **do**
      if $i \in S_t$ **then**
        ClientUpdate($\mathbf{u}_{i,t}$, i)
      else
        $\mathbf{z}_{i,t+1} \leftarrow \mathbf{z}_{i,t}$
      end
    end
    $\bar{\boldsymbol{\Theta}} \leftarrow \mathsf{P}_H(\mathbf{z}_{t+1})$ `// over` $\mathbf{z}_{t+1}$ `of present clients`
    $\boldsymbol{\Theta}_{t+1} = (1 - \beta_t)\mathbf{z}_{t+1} + \beta_t \bar{\boldsymbol{\Theta}}$
    $\mathbf{u}_{t+1} = (1 - \gamma_t)\mathbf{u}_t + \gamma_t \boldsymbol{\Theta}_{t+1}$
  end
ClientUpdate *(*$\mathbf{u}_{i,t}$, i*)*:
  compute $\mathsf{P}_{f_i}^{\eta_t}(\mathbf{u}_{i,t})$ `// using standard gradient descent`
  $\mathbf{z}_{i,t+1} \leftarrow (1 - \alpha_t)\mathbf{u}_{i,t} + \alpha_t \mathsf{P}_{f_i}^{\eta_t}(\mathbf{u}_{i,t})$

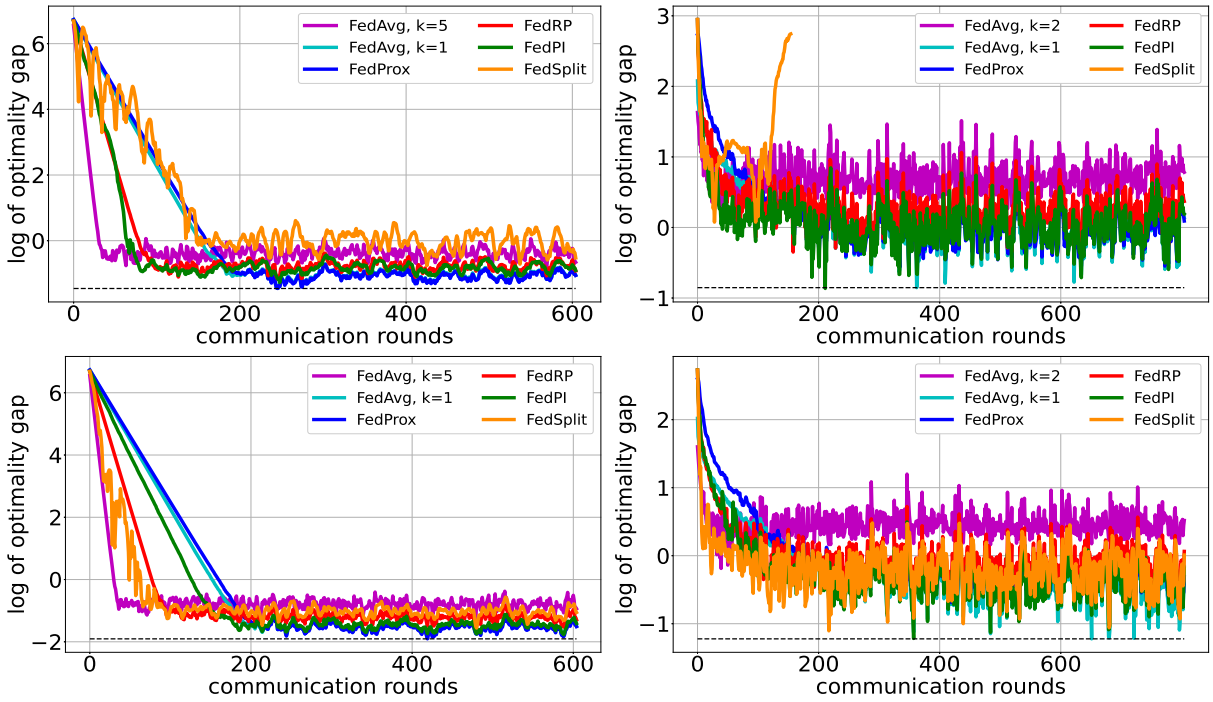


Figure B.1: Effect of client sampling. Top-Left: least squares with $p = 0.5$ (participation probability); Top-Right: logistic regression with $p = 0.5$; Bottom-Left: least squares with $p = 0.7$; Bottom-Right: logistic regression with $p = 0.7$.

B.2 Effect of using mini-batch gradient descent locally

Figure B.2 shows the results when mini-batch gradient descent is used locally by clients (for different splitting algorithms), as in Algorithm 4. As in the grand scheme (2.15)-(2.17), we assume full client participation. As observed in Figure B.2, with mini-batch gradient descent used locally, none of the aforementioned algorithms converge to the correct solution even in convex settings. Furthermore, convergence speed of the algorithms decreases compared to when clients use standard gradient descent. Also, FedAvg , even with $k = 1$, does not outperform the other splitting algorithms, and FedSplit and FedPi converge to better solutions. On the other hand, in the nonconvex setting, FedAvg with $k = 100$ achieves better solutions.

Algorithm 4: Mini-batch Gradient Descent for local computations.

Input: B (local minibatch size), $\alpha_t, \beta_t, \gamma_t, \eta_t$
Output: $\bar{\mathbf{\Theta}}$ (final global model)
initialize $\mathbf{u}_0$ for each client i
Server:
  for *each round* $t = 1, 2, \dots$ **do**
    for *each client* i ***in parallel*** **do**
      ClientUpdate$(\mathbf{u}_{i,t}, i)$
    end
    $\bar{\mathbf{\Theta}} \leftarrow \mathsf{P}_H(\mathbf{z}_{t+1})$
    $\mathbf{\Theta}_{t+1} = (1-\beta_t)\mathbf{z}_{t+1} + \beta_t \bar{\mathbf{\Theta}}$
    $\mathbf{u}_{t+1} = (1-\gamma_t)\mathbf{u}_t + \gamma_t \mathbf{\Theta}_{t+1}$
  end
ClientUpdate $(\mathbf{u}_{i,t}, i)$**:**
  compute $\mathsf{P}_{f_i}^{\eta_t}(\mathbf{u}_{i,t})$ `// local update with batch size` B
  $\mathbf{z}_{i,t+1} \leftarrow (1-\alpha_t)\mathbf{u}_{i,t} + \alpha_t \mathsf{P}_{f_i}^{\eta_t}(\mathbf{u}_{i,t})$

B.3 Results with error bars for Figure 2.8, Right

Implementation details of Anderson acceleration for nonconvex CNN: In practice, we found that applying the acceleration algorithm to $\mathsf{P}_H(\mathbf{z}_{t+1})$ in (2.16) rather than $\mathbf{u}_{t+1}$ in (2.17) leads to a more stable algorithm. So, we applied acceleration to $\mathsf{P}_H(\mathbf{z}_{t+1})$, and used the result to find $\mathbf{u}_{t+1}$ and $\mathbf{\Theta}_{t+1}$ for the nonconvex model. Figure B.3 shows the effect of Anderson acceleration on the nonconvex CNN with memory size $\tau = 10$. FedRP and FedSplit are less stable and Anderson acceleration actually made them worse while FedAvg , FedProx and FedPi converge much faster with our implementation of Anderson acceleration. The deviation among 4 different runs is shown as the shaded area.

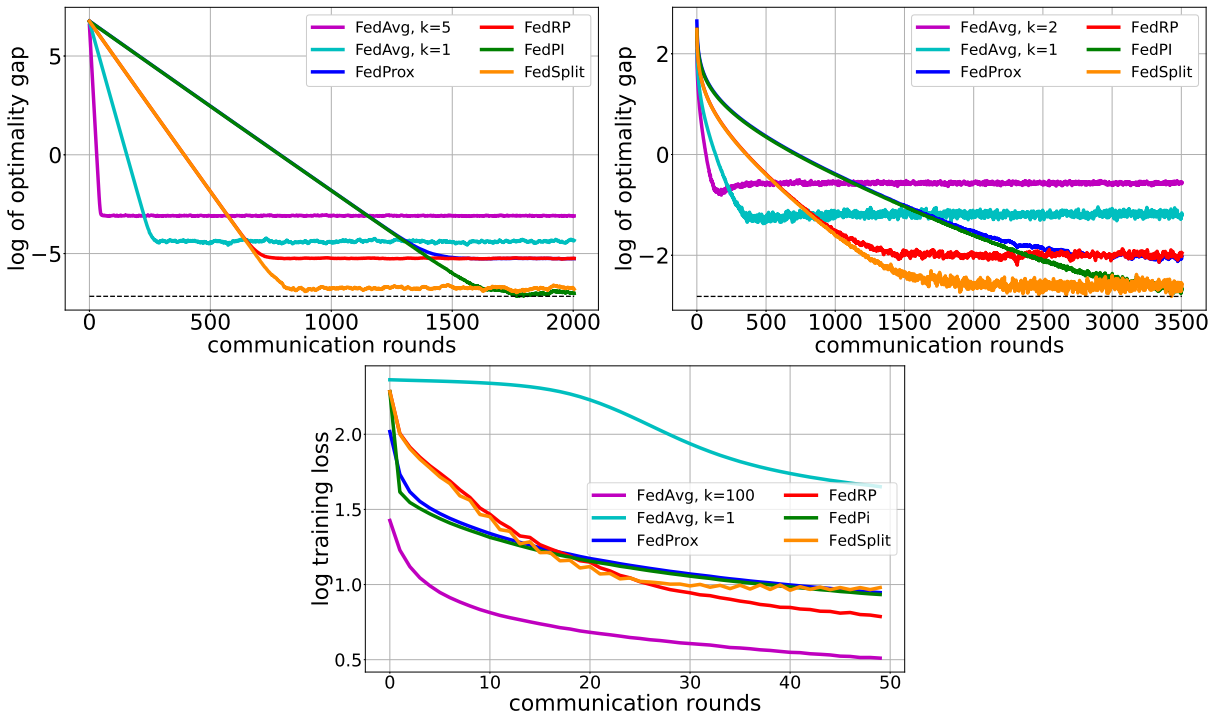


Figure B.2: The effect of using minibatch gradient descent. Local datasets are divided into 5 minibatches. Top-Left: least squares with $B = 1000$. Top-Right: logistic regression with $B = 200$. Bottom: nonconvex CNN on MNIST with $B = 500$.

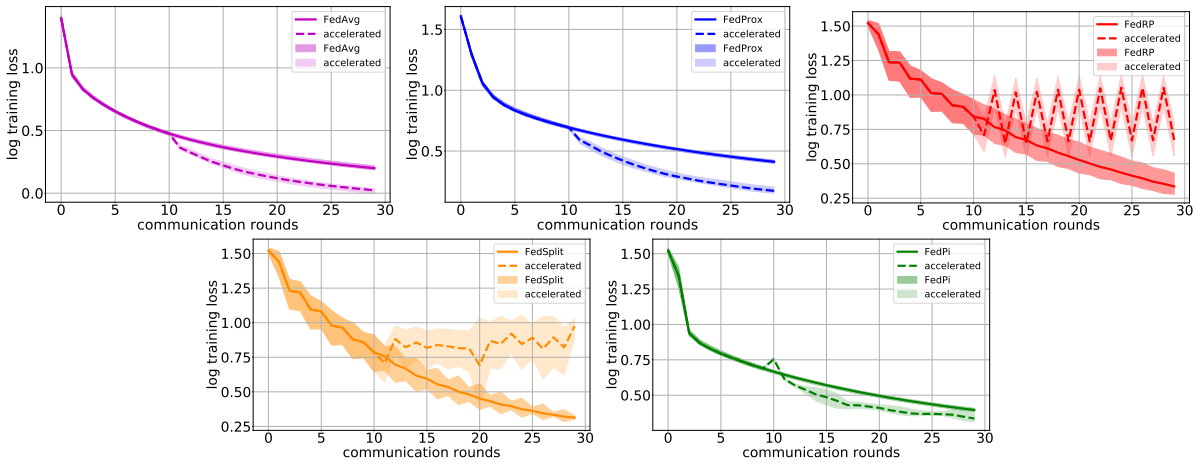


Figure B.3: Detailed results for Anderson acceleration for nonconvex optimization problem of CNN training with $\tau = 10$. Dashed lines are the accelerated results.

C Proofs

We first recall the following convenient result:

Theorem 12 ((Brézis and Browder 1976), (Bauschke and Combettes 2017, Theorem 5.36)). *Let $\{\boldsymbol{\theta}_t\}$ and $\{\mathbf{z}_t\}$ be two sequences in $\mathbb{R}^p$, $\emptyset \neq \mathsf{F} \subseteq \mathbb{R}^p$, and $W_k := \operatorname{cl}\operatorname{conv} \boldsymbol{\theta}_t : t \geq k$. Suppose that*

1. *for all $\boldsymbol{\theta} \in \mathsf{F}$, $\|\boldsymbol{\theta}_t - \boldsymbol{\theta}\|_2^2 \to p(\boldsymbol{\theta}) < \infty$ $\big($i.e. $\{\boldsymbol{\theta}_t\}$ is quasi-Fejér monotone w.r.t. $\mathsf{F}\big)$;*
2. *for all k, $\operatorname{dist}(\mathbf{z}_t, W_k) \to 0$ as $t \to \infty$.*

Then, the sequence $\{\mathbf{z}_t\}$ has at most one limit point in F. Therefore, if additionally

3. *all limit points of $\{\mathbf{z}_t\}$ lie in F,*

then the whole sequence $\{\mathbf{z}_t\}$ converges to a point in F.

Below, we use well-known properties about firm nonexpansions and Fejér monotone sequences, see the excellent book (Bauschke and Combettes 2017) for background.

Theorem 2. *Assuming each client participates indefinitely, the step size η_t is bounded from below (i.e. $\liminf_t \eta_t > 0$), the client functions $\{f_i\}$ are convex, and homogeneous in the sense that they have a common minimizer, i.e. $\mathsf{F} := \bigcap_i \operatorname{argmin}_{\boldsymbol{\theta}_i} f_i(\boldsymbol{\theta}_i) \neq \emptyset$, then the iterates of* FedProx *converge to a point in F, which is a solution of (2.1).*

Proof. The proof is a simplification and application of (Tseng 1992).

Consider all possible combinations of client participation, i.e. let $\mathcal{I} := 2^{[n]} \setminus \emptyset$ be the power set of clients $[n] := \{1, \dots, n\}$, with the empty set removed. For each $I \in \mathcal{I}$, define

$$\mathsf{T}_I = \mathsf{T}_{I,\eta} := \frac{1}{|I|} \sum_{i \in I} \mathsf{P}_{f_i}^{\eta}.$$

For any $\boldsymbol{\theta} \in \mathsf{F}$, we have $\boldsymbol{\theta} = \mathsf{P}_{f_i}^{\eta} \boldsymbol{\theta}$ for all i, and hence $\mathsf{F} \subseteq \cap_{I \in \mathcal{I}} \operatorname{Fix} \mathsf{T}_I$ (equality in fact when we consider singleton I). Moreover, since we assume each f_i to be convex, the mapping $\mathsf{T}_{I,\eta}$ is firmly nonexpansive. In fact, $\mathsf{T}_{I,\eta} = \mathsf{P}_{f_I}^{\eta}$ for a so-called proximal average function f_I (Bauschke and Combettes 2017), and hence it follows from (Tseng 2000) that

$$(1 \wedge \eta)\|\boldsymbol{\theta} - \mathsf{T}_{I,1}\boldsymbol{\theta}\|_2 \leq \|\boldsymbol{\theta} - \mathsf{T}_{I,\eta}\boldsymbol{\theta}\|_2.$$

Since we assume $\liminf_t \eta_t := \eta > 0$, the aforementioned monotonicity allows us to assume w.l.o.g. that $\eta_t \equiv \eta > 0$.

With this established notation we rewrite FedProx simply as

$$\boldsymbol{\theta}_{t+1} = \mathsf{T}_{I_t} \boldsymbol{\theta}_t,$$

where $I_t \in \mathcal{I}$ can be arbitrary, except for all i: $|\{t : i \in I_t\}| = \infty$.

Applying the firm nonexpansiveness of T_{I_t}, we have for any $\boldsymbol{\theta} \in \mathsf{F}$:

$$\|\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}\|_2^2 + \|\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t+1}\|_2^2 \leq \|\boldsymbol{\theta}_t - \boldsymbol{\theta}\|_2^2,$$

whence

$$\|\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}\|_2^2 \leq \|\boldsymbol{\theta}_t - \boldsymbol{\theta}\|_2^2,$$

i.e. $\{\boldsymbol{\theta}_t\}$ is Fejér monotone w.r.t. F. Moreover,

$$\boldsymbol{\theta}_t - \mathsf{T}_{I_t}\boldsymbol{\theta}_t \to \mathbf{0}. \tag{A.1}$$

Since $\{\boldsymbol{\theta}_t\}$ is bounded and there are only finitely many choices for $I_t \in \mathcal{I}$, from (A.1) there exists $\mathbf{z} \in \cap_{J\in\mathcal{J}}\mathrm{Fix}\mathsf{T}_J$ that is a limit point of $\{\boldsymbol{\theta}_t\}$ for some $\emptyset \neq \mathcal{J} \subseteq \mathcal{I}$. Take a subsequence $\boldsymbol{\theta}_{t_k} \to \mathbf{z}$ and let $s_k = \min\{t \geq t_k : I_t \notin \mathcal{J}\}$. Pass to a subsequence we may assume $I_{s_k} \equiv I$ and $\boldsymbol{\theta}_{s_k} \to \boldsymbol{\theta}$. Since $\boldsymbol{\theta}_{s_k} - \mathsf{T}_I\boldsymbol{\theta}_{s_k} \to \mathbf{0}$ (see (A.1)) we have $\boldsymbol{\theta} \in \mathrm{Fix}\mathsf{T}_I$. Since $\mathbf{z} \in \cap_{J\in\mathcal{J}}\mathrm{Fix}\mathsf{T}_J$ and $I_t \in \mathcal{J}$ for $t \in [t_k, s_k)$ we have

$$\|\boldsymbol{\theta}_{s_k} - \mathbf{z}\|_2 \leq \|\boldsymbol{\theta}_{t_k} - \mathbf{z}\|_2 \to 0,$$

and hence $\mathbf{z} = \boldsymbol{\theta} \in \mathrm{Fix}\mathsf{T}_I = \cap_{i\in I}\mathrm{Fix}\mathsf{P}_{f_i}$. Since each client f_i participates infinitely often, we may continue the argument to conclude that any limit point $\mathbf{z} \in \mathsf{F}$. Applying Theorem 12 we know the whole sequence $\boldsymbol{\theta}_t \to \boldsymbol{\theta}_\infty \in \mathsf{F}$. □

We remark that the convexity condition on f_i may be further relaxed using arguments in (Aleyner and Reich 2009).

Theorem 3 ((Lions 1978; Passty 1979))**.** *Assuming each client participates in every round, the step size η_t satisfies $\sum_t \eta_t = \infty$ and $\sum_t \eta_t^2 < \infty$, and the functions $\{f_i\}$ are convex, then the* averaged *iterates $\bar{\boldsymbol{\Theta}}_t := \frac{\sum_{s=1}^t \eta_s \boldsymbol{\Theta}_s}{\sum_{s=1}^t \eta_s}$ of* FedProx *converge to a solution of the* original *problem* (2.1).

Proof. We simply verify Theorem 12. Let $\boldsymbol{\Theta} \in \mathrm{dom}\,\mathsf{f} \cap H$, $\mathbf{a}^* \in \partial\mathsf{f}(\boldsymbol{\Theta})$ and $\mathbf{b}^* \in H^\perp$. Applying the firm nonexpansiveness of $\mathsf{P}_\mathsf{f}^{\eta_t}$:

$$\begin{aligned}
\|\mathsf{P}_\mathsf{f}^{\eta_t}\boldsymbol{\Theta}_t - \boldsymbol{\Theta}\|_2^2 &= \|\mathsf{P}_\mathsf{f}^{\eta_t}\boldsymbol{\Theta}_t - \mathsf{P}_\mathsf{f}^{\eta_t}(\boldsymbol{\Theta} + \eta_t\mathbf{a}^*)\|_2^2 \\
&\leq \|\boldsymbol{\Theta}_t - \boldsymbol{\Theta} - \eta_t\mathbf{a}^*\|_2^2 - \|\boldsymbol{\Theta}_t - \mathsf{P}_\mathsf{f}^{\eta_t}\boldsymbol{\Theta}_t - \eta_t\mathbf{a}^*\|_2^2 \\
&= \|\boldsymbol{\Theta}_t - \boldsymbol{\Theta}\|_2^2 - \|\boldsymbol{\Theta}_t - \mathsf{P}_\mathsf{f}^{\eta_t}\boldsymbol{\Theta}_t\|_2^2 + 2\eta_t\left\langle\boldsymbol{\Theta} - \mathsf{P}_\mathsf{f}^{\eta_t}\boldsymbol{\Theta}_t; \mathbf{a}^*\right\rangle, \\
\|\mathsf{P}_H\mathsf{P}_\mathsf{f}^{\eta_t}\boldsymbol{\Theta}_t - \boldsymbol{\Theta}\|_2^2 &\leq \|\mathsf{P}_\mathsf{f}^{\eta_t}\boldsymbol{\Theta}_t - \boldsymbol{\Theta}\|_2^2 - \|\mathsf{P}_\mathsf{f}^{\eta_t}\boldsymbol{\Theta}_t - \mathsf{P}_H\mathsf{P}_\mathsf{f}^{\eta_t}\boldsymbol{\Theta}_t\|_2^2 + 2\eta_t\left\langle\boldsymbol{\Theta} - \mathsf{P}_H\mathsf{P}_\mathsf{f}^{\eta_t}\boldsymbol{\Theta}_t; \mathbf{b}^*\right\rangle.
\end{aligned}$$

Summing the above two inequalities and applying the inequality $-\|\mathbf{x}\|_2^2 + 2\left\langle\mathbf{x};\mathbf{y}\right\rangle \leq \|\mathbf{y}\|_2^2$ repeatedly:

$$\|\mathsf{P}_H\mathsf{P}_\mathsf{f}^{\eta_t}\boldsymbol{\Theta}_t - \boldsymbol{\Theta}\|_2^2 \leq \|\boldsymbol{\Theta}_t - \boldsymbol{\Theta}\|_2^2 + 2\eta_t\left\langle\boldsymbol{\Theta} - \boldsymbol{\Theta}_t; \mathbf{a}^* + \mathbf{b}^*\right\rangle + \eta_t^2[\|\mathbf{a}^* + \mathbf{b}^*\|_2^2 + \|\mathbf{a}^*\|_2^2]. \tag{A.2}$$

Summing over t and rearranging we obtain for any $\boldsymbol{\Theta} \in \operatorname{dom} \mathsf{f} \cap H$, $\boldsymbol{\Theta}^* = \mathbf{a}^* + \mathbf{b}^*$:

$$2\left\langle \boldsymbol{\Theta} - \bar{\boldsymbol{\Theta}}_t; \boldsymbol{\Theta}^* \right\rangle + [\|\mathbf{a}^*\|_2^2 + \|\boldsymbol{\Theta}^*\|_2^2] \sum_{k=0}^{t} \eta_k^2/\Lambda_t \geq (\|\boldsymbol{\Theta}_{t+1} - \boldsymbol{\Theta}\|_2^2 - \|\boldsymbol{\Theta}_1 - \boldsymbol{\Theta}\|_2^2)/\Lambda_t,$$

where $\Lambda_t := \sum_{k=1}^{t} \eta_k$. Using the assumptions on η_t we thus know

$$\liminf_{t\to\infty} \left\langle \boldsymbol{\Theta} - \bar{\boldsymbol{\Theta}}_t; \boldsymbol{\Theta}^* \right\rangle \geq 0.$$

Since $\boldsymbol{\Theta} \in \operatorname{dom} \mathsf{f} \cap H$ is arbitrary and $\boldsymbol{\Theta}^*$ is chosen to be its subdifferential, it follows that any limit point of $\{\bar{\boldsymbol{\Theta}}_t\}$ is in F (the set of solutions of the original problem (2.1)), i.e. condition 3 of Theorem 12 holds. If $\{\bar{\boldsymbol{\Theta}}_t\}$ is bounded, which we assume now, then $\mathsf{F} \neq \emptyset$. Let $\boldsymbol{\Theta} \in \mathsf{F}$ and set $\boldsymbol{\Theta}^* = \mathbf{0}$ we know from (A.2) that $\{\boldsymbol{\Theta}_t\}$ is quasi-Fejér monotone w.r.t. F (i.e. condition 1 in Theorem 12 holds). Lastly, let $\bar{\eta}_{t,k} := \eta_k/\Lambda_t$ and we verify condition (II) in Theorem 12:

$$\begin{aligned}
\operatorname{dist}(\bar{\boldsymbol{\Theta}}_t, W_k) &\leq \left\| \sum_{\kappa=1}^{t} \bar{\eta}_{t,\kappa} \boldsymbol{\Theta}_\kappa - \sum_{s=k}^{t} \bar{\eta}_{t,s} \boldsymbol{\Theta}_s \Big/ \sum_{s=k}^{t} \bar{\eta}_{t,s} \right\|_2 \\
&\leq \sum_{\kappa=1}^{k-1} \bar{\eta}_{t,\kappa} \left[\|\boldsymbol{\Theta}_\kappa\|_2 + \left\| \sum_{s=k}^{t} \bar{\eta}_{t,s} \boldsymbol{\Theta}_s \right\|_2 \Big/ \sum_{s=k}^{t} \bar{\eta}_{t,s} \right] \\
&\xrightarrow{t\to\infty} 0,
\end{aligned}$$

since $\boldsymbol{\Theta}_t$ is bounded and for any k, $\bar{\eta}_{t,k} \to 0$ as $t \to \infty$. All three conditions in Theorem 12 are now verified. Therefore, the whole sequence $\{\bar{\boldsymbol{\Theta}}_t\}$ converges to a point in F. □

We remark that the step size condition is tight, as shown by the following simple example:

Example 1. *Let* $f_\pm(w) = \frac{1}{2}(w \pm 1)^2$. *Simple calculation verifies that*

$$\mathsf{P}^{\eta}_{f_\pm}(w) = \tfrac{w \mp \eta}{1+\eta}.$$

Therefore, the iterates of FedProx *for the two functions* f_+ *and* f_- *are:*

$$w_{t+1} = \tfrac{w_t}{1+\eta_t} = \prod_{\tau=0}^{t} \tfrac{1}{1+\eta_\tau} w_0,$$

which tends to the true minimizer $w^\star = 0$ *for any* w_0 *iff*

$$\prod_{\tau=0}^{t} \tfrac{1}{1+\eta_\tau} \to 0 \iff \sum_t \eta_t \to \infty.$$

If we consider instead f_+ *and* $2f_-$, *then*

$$2w_{t+1} = \tfrac{w_t - \eta_t}{1+\eta_t} + \tfrac{w_t + 2\eta_t}{1+2\eta_t}.$$

Passing to a subsequence if necessary, let $\eta_t \to \eta \neq 0$ and suppose $w_t \to w^\star = \frac{1}{3}$, then passing to the limit we obtain

$$\begin{aligned} 2 = \tfrac{1-3\eta}{1+\eta} + \tfrac{1+6\eta}{1+2\eta} &\iff 2(1+\eta)(1+2\eta) = (1-3\eta)(1+2\eta) + (1+6\eta)(1+\eta) \\ &\iff \eta = 0, \end{aligned}$$

contradiction. Therefore, it is necessary to have $\eta_t \to 0$ in order for FedProx *to converge to the true solution $w^\star = \frac{1}{3}$ on this example.*

Theorem 6. *If the reflector R_f^η is a (strict) contraction, then f must be strongly convex.*

Proof. Since $\mathsf{R}_f^\eta = \mathsf{R}_{\eta f}$ and f is strongly convex iff ηf is so, w.l.o.g. we may take $\eta = 1$. Suppose R_f is γ-contractive for some $\gamma \in (0,1)$, i.e. for all $\boldsymbol{\theta}$ and $\mathbf{z}$:

$$\|\mathsf{R}_f\boldsymbol{\theta} - \mathsf{R}_f\mathbf{z}\|_2 \leq \gamma \cdot \|\boldsymbol{\theta} - \mathbf{z}\|_2.$$

It then follows that the proximal map $\mathsf{P}_f = \frac{\mathrm{Id}+\mathsf{R}_f}{2}$ is $\frac{1+\gamma}{2}$-contractive. Moreover, $\frac{2}{1+\gamma}\mathsf{P}_f$, being nonexpansive, is the gradient of the convex function $\frac{2}{1+\gamma}\mathsf{M}_f$. Thus, $\frac{2}{1+\gamma}\mathsf{P}_f$ is actually firmly nonexpansive (Bauschke and Combettes 2017, Corollary 18.17). But,

$$\tfrac{2}{1+\gamma}\mathsf{P}_f = \tfrac{2}{1+\gamma}(\mathrm{Id} + \partial f)^{-1} = [\mathrm{Id} + (\partial f - \tfrac{1-\gamma}{1+\gamma}\mathrm{Id}) \circ \tfrac{1+\gamma}{2}\mathrm{Id}]^{-1},$$

and hence $(\partial f - \frac{1-\gamma}{1+\gamma}\mathrm{Id}) \circ \frac{1+\gamma}{2}\mathrm{Id}$ is maximal monotone (Bauschke and Combettes 2017, Proposition 23.8), i.e. f is $\frac{1-\gamma}{1+\gamma}$-strongly convex. □

Theorem 8. *Let each client participate in every round and the functions $\{f_i\}$ be convex. If the step size $\eta_t \equiv \eta$ is constant, then any fixed point of* FedRP *is a solution of the* regularized *problem* (2.6)*. If the reflector $\mathsf{R}_\mathsf{f}^\eta$ is idempotent (i.e. $\mathsf{R}_\mathsf{f}^\eta\mathsf{R}_\mathsf{f}^\eta = \mathsf{R}_\mathsf{f}^\eta$) and the clients are homogeneous, then the vanilla iterates $\boldsymbol{\Theta}_t$ of* FedRP *converge.*

Proof. To see the first claim, let $\boldsymbol{\Theta}$ be a fixed point of FedRP , i.e.

$$\boldsymbol{\Theta} = \mathsf{P}_H\mathsf{R}_\mathsf{f}^\eta\boldsymbol{\Theta} = \mathsf{P}_H(2\mathsf{P}_\mathsf{f}^\eta\boldsymbol{\Theta} - \boldsymbol{\Theta}) = 2\mathsf{P}_H\mathsf{P}_\mathsf{f}^\eta\boldsymbol{\Theta} - \mathsf{P}_H\boldsymbol{\Theta} = 2\mathsf{P}_H\mathsf{P}_\mathsf{f}^\eta\boldsymbol{\Theta} - \boldsymbol{\Theta},$$

since P_H is linear and $\boldsymbol{\Theta} \in H$ due to the projection. Thus, $\boldsymbol{\Theta} = \mathsf{P}_H\mathsf{P}_\mathsf{f}^\eta\boldsymbol{\Theta}$. In other words, the fixed points of FedRP are exactly those of FedProx , and the first claim then follows from (Bauschke et al. 2005), see the discussions in Section 2.4.2 and also (Bauschke and Kruk 2004, Lemma 7.1).

To prove the second claim, we first observe that $\{\boldsymbol{\Theta}_t\}$ is Fejér monotone w.r.t. F, the solution set of the regularized problem (2.6). Indeed, for any $\boldsymbol{\Theta} \in \mathsf{F}$, using the firm nonexpansiveness of P_H we have

$$\|\boldsymbol{\Theta}_{t+1} - \boldsymbol{\Theta}\|_2^2 + \|\mathsf{R}_\mathsf{f}^\eta\boldsymbol{\Theta}_t - \boldsymbol{\Theta}_{t+1} - \mathsf{R}_\mathsf{f}^\eta\boldsymbol{\Theta} + \boldsymbol{\Theta}\|_2^2 \leq \|\mathsf{R}_\mathsf{f}^\eta\boldsymbol{\Theta}_t - \mathsf{R}_\mathsf{f}^\eta\boldsymbol{\Theta}\|_2^2 \leq \|\boldsymbol{\Theta}_t - \boldsymbol{\Theta}\|_2^2.$$

Summing and telescoping the above we know

$$\mathsf{R}_{\mathsf{f}}^{\eta}\boldsymbol{\Theta}_t - \boldsymbol{\Theta}_{t+1} \to \mathsf{R}_{\mathsf{f}}^{\eta}\boldsymbol{\Theta} - \boldsymbol{\Theta}.$$

Let $\boldsymbol{\Theta}_\infty$ be a limit point of $\{\boldsymbol{\Theta}_t\}$ (which exists since $\boldsymbol{\Theta}_t$ is bounded). Since $\mathsf{R}_{\mathsf{f}}^{\eta}$ is idempotent, we have the range $\operatorname{rge}\mathsf{R}_{\mathsf{f}}^{\eta}$ equal to its fixed point $\operatorname{Fix}\mathsf{R}_{\mathsf{f}}^{\eta}$. When the clients are homogeneous, we can choose $\boldsymbol{\Theta} = \mathsf{R}_{\mathsf{f}}^{\eta}\boldsymbol{\Theta}$. Therefore, $\boldsymbol{\Theta}_\infty \in \operatorname{rge}\mathsf{R}_{\mathsf{f}} = \operatorname{Fix}\mathsf{R}_{\mathsf{f}}$. Since $\boldsymbol{\theta}_\infty \in H$ by definition, $\boldsymbol{\theta}_\infty \in \mathsf{F}$. Therefore, all limit points of $\{\boldsymbol{\Theta}_t\}$ are in F. Applying Theorem 12, we conclude that the entire Fejér sequence $\{\boldsymbol{\Theta}_t\}$ converges to $\boldsymbol{\theta}_\infty \in \mathsf{F}$. □

As shown in (Bauschke and Kruk 2004), a (closed) convex cone is obtuse iff its reflector is idempotent. To give another example: the univariate, idempotent reflector $\mathsf{R}_f(w) = (w)_+ := \max\{w, 0\}$ leads to

$$\mathsf{P}_f(w) = \tfrac{w+\mathsf{R}_f w}{2} = \tfrac{w+(w)_+}{2} \implies f(w) = \tfrac{1}{2}(-w)_+^2.$$

Interestingly, the epigraph of the above f is a convex set but not a cone and yet its reflector is idempotent. Therefore, the idempotent condition introduced here is a strict generalization of (Bauschke and Kruk 2004).

Appendix B

Appendix of Chapter 3

A Experimental setup

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

A.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. 2017b). 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 B.1. To update the local models at each user using its local data, unless otherwise is stated, we apply stochastic gradient descent (SGD).

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. 2017b): 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 $p = 21,272,778$

Table B.1: 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		

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 B.2: 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 B.1.

Datasets	**Train set size**	**Test set size**	**Data Partition method**	**# of clients**	**Model**	**# of parameters**
MNIST	48000	12000	sharding (McMahan et al. 2017b)	20/40/60	CNN (Table B.1)	28,938
FMNIST	50000	10000	sharding (McMahan et al. 2017b)	20	CNN (Table B.1)	28,938
CIFAR10	50000	10000	sharding (McMahan et al. 2017b)	20	ResNet-18 (He et al. 2015)	11,181,642
CIFAR100	50000	10000	sharding (McMahan et al. 2017b)	20	ResNet-34 (He et al. 2015)	21,272,778

A.2 DP training parameters

For each dataset, we sample the privacy parameter ϵ of clients from different distributions, as shown in Table 3.3. 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 $10^{-4} < N_i^{-1}$ (for all i). We also set the clipping threshold c equal to 3.

A.3 Algorithms to compare and tuning hyperparameters

We compare our Robust-HDP, which benefits from RPCA (Algorithm 6), with four baseline algorithms, including WeiAvg (Liu et al. 2021a) (Algorithm 5), 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 B.3 to Table B.6.

Algorithm 5: WeiAvg (Liu et al. 2021a)

Input: Initial parameter $\boldsymbol{\theta}^0$, Clients batch sizes $\{b_1, \ldots, b_n\}$, Clients dataset sizes $\{N_1, \ldots, N_n\}$,
Clients noise scales $\{z_1, \ldots, z_n\}$, gradient norm bound c, local epochs $\{K_1, \ldots, K_n\}$, global round E,
privacy parameter δ, number of model parameters p, privacy accountant **PA**.
Output: $\boldsymbol{\theta}_E, \{\epsilon_1, \ldots, \epsilon_n\}$

Initialize $\boldsymbol{\theta}_0$ randomly.
for $e \in [E]$ **do**
  sample a set of clients $\mathcal{S}^e \subseteq \{1, \ldots, n\}$
  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**$(\frac{b_i}{N_i}, z_i, K_i, e)$
  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: $\boldsymbol{\theta}^E, \{\epsilon_1^E, \ldots, \epsilon_n^E\}$

Algorithm 6: Principal Component Pursuit by Alternating Directions (Candès 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 B.3: The learning rates used for training with each algorithm on MNIST dataset

alg \ dist	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 B.4: The learning rates used for training with each algorithm on FMNIST dataset

alg \ dist	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 B.5: The learning rates used for training with each algorithm on CIFAR10 dataset

alg \ dist	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

B 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:

Table B.6: The learning rates used for training with each algorithm on CIFAR100 dataset.

alg \ dist	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

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

We also have:

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

where the last equation has used Equation (3.2) and that we clip the norm of sample gradients $\bar{g}_{ij}(\boldsymbol{\theta})$ with an "effective" clipping threshold c. We can now plug Equation (B.2) into the parenthesis in Equation (B.1) and rewrite it as:

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

$$
\begin{aligned}
&= \frac{1}{b_i^2}\left(b_i c^2 + \left(2\binom{b_i}{2} - b_i^2\right)\left\|G_i(\boldsymbol{\theta})\right\|^2\right) + \frac{pc^2z^2(\epsilon_i,\delta_i,q_i,K_i,E)}{b_i^2} \\
&= \frac{1}{b_i^2}\left(b_i c^2 - b_i\left\|G_i(\boldsymbol{\theta})\right\|^2\right) + \frac{pc^2z^2(\epsilon_i,\delta_i,q_i,K_i,E)}{b_i^2} \\
&= \frac{c^2 - \left\|G_i(\boldsymbol{\theta})\right\|^2}{b_i} + \frac{pc^2z^2(\epsilon_i,\delta_i,q_i,K_i,E)}{b_i^2} \approx \frac{pc^2z^2(\epsilon_i,\delta_i,q_i,K_i,E)}{b_i^2}
\end{aligned}
$$

C Used lemmas

In this section, we state some lemmas, which we will use in our proofs.

Lemma 6 (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:

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

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

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

□

Lemma 7. *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:*

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

Proof. From the definition of variance, we have:

$$
\begin{aligned}
\mathbb{E}[\|\sum_{i=1}^n v_i\|^2] &= \|\sum_{i=1}^n \mathcal{E}_i\|^2 + \mathbb{E}[\|\sum_{i=1}^n (v_i - \mathcal{E}_i)\|^2] \\
&\leq \|\sum_{i=1}^n \mathcal{E}_i\|^2 + n\sum_{i=1}^n \mathbb{E}[\|v_i - \mathcal{E}_i\|^2]
\end{aligned}
$$

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

where the inequality is based on the Lemma 6. □

Property 1 (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)-DP and their domains $\mathcal{D}_i$ are disjoint subsets. Any function g of the form $g(M_1, \ldots, M_n)$ satisfies $(\max_i \epsilon_i, \max_i \delta_i)$-DP.*

D Proofs

D.1 Proof of Theorem 9

Theorem 9. *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 *provides $(\epsilon_i, \delta_i)-DP$ to client 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. *Also, assuming clients' datasets are disjoint, the algorithm also satisfies $(\epsilon_{max}, \delta_{max})$-DP, where $(\epsilon_{max}, \delta_{max}) = \big(\max(\{\epsilon_i\}_{i=1}^n), \max(\{\delta_i\}_{i=1}^n)\big)$.*

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)-DP independently. Hence, it satisfies heterogeneous 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 $\big(\max(\{\epsilon_i\}_{i=1}^n), \max(\{\delta_i\}_{i=1}^n)\big)$-DP, according to the parallel composition property above. □

D.2 Proof of Theorem 10

Theorem 10 (Robust-HDP). *Assume that Assumptions 1 and 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:*

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

where $E_l^{min} = \min_i E_i$, i.e. the minimum number of local DPSGD steps per round across clients. Also, Ψ_p and Ψ_σ are two constants controlling the quality of the final model parameter returned by Robust-HDP *explained below. Also, assuming a large E, $\min_{0 \leq e \leq E-1} \mathbb{E}[\|\nabla f(\boldsymbol{\theta}^e)\|^2]$ happens close to the convergence time, i.e. at the last communication rounds.*

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

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

Therefore:

$$\begin{aligned}
&\mathbb{E}[\tilde{g}_i(\boldsymbol{\theta})] = \mathbb{E}[g_i(\boldsymbol{\theta})] = \nabla f_i(\boldsymbol{\theta}) \\
&\texttt{Var}(\tilde{g}_i(\boldsymbol{\theta})) = \texttt{Var}(g_i(\boldsymbol{\theta})) + \frac{p\sigma_{i,\texttt{DP}}^2}{b_i^2} \leq \sigma_{i,\tilde{g}}^2 := \sigma_{i,g}^2 + \frac{p\sigma_{i,\texttt{DP}}^2}{b_i^2}.
\end{aligned} \tag{B.3}$$

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 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:

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

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:

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

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:

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

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

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

Hence,

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

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:

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

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

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

$$+\frac{\beta\eta_l^2}{2}\mathbb{E}\Big[\Big\|\sum_{i=1}^{n} w_i^e \sum_{k=0}^{E_i-1}\big(\tilde{g}_i(\boldsymbol{\theta}_{i,k}^e)-\nabla f(\boldsymbol{\theta}^e)\big)\Big\|^2\Big].$$

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

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

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: $\eta_l \leq \frac{1}{6\beta\bar{E}_l^e}$):

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

Therefore,

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

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

$$\mathbb{E}\big[f(\boldsymbol{\theta}^{e+1})\big] \leq \mathbb{E}\big[f(\boldsymbol{\theta}^e)\big]-\eta_l\frac{11\bar{E}_l^e-6}{12}\mathbb{E}\big[\|\nabla f(\boldsymbol{\theta}^e)\|^2\big]$$

$$+\frac{1}{2}\eta_l(1-\beta\eta_l\bar{E}_l^e)\underbrace{\mathbb{E}\left[\left\|\sum_{i=1}^{n}w_i^e\sum_{k=0}^{E_i-1}\left(\nabla f_i(\boldsymbol{\theta}_{i,k}^e)-\nabla f(\boldsymbol{\theta}^e)\right)\right\|^2\right]}_{\mathcal{B}}$$
$$+\beta\eta_l^2\underbrace{\mathbb{E}\left[\left\|\sum_{i=1}^{n}w_i^e\sum_{k=0}^{E_i-1}\left(\tilde{g}_i(\boldsymbol{\theta}_{i,k}^e)-\nabla f_i(\boldsymbol{\theta}_{i,k}^e)\right)\right\|^2\right]}_{\mathcal{A}}$$
$$+\beta\eta_l^2\underbrace{\mathbb{E}\left[\left\|\sum_{i=1}^{n}w_i^e\sum_{k=0}^{E_i-1}\left(\nabla f_i(\boldsymbol{\theta}_{i,k}^e)-\nabla f(\boldsymbol{\theta}^e)\right)\right\|^2\right]}_{\mathcal{B}} \tag{B.4}$$

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

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

where in the first and second inequalities, we used Cauchy-Schwarz inequality. In the last inequality, we used Equation (B.3). Similarly, we can bound $\mathcal{B}$:

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

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:

$$\mathcal{B}=\mathbb{E}\left[\left\|\left(\sum_{i=1}^{n}\Delta_i^e\sum_{k=0}^{E_i-1}\nabla f_i(\boldsymbol{\theta}_{i,k}^e)\right)+\left(\sum_{i=1}^{n}\lambda_i\sum_{k=0}^{E_i-1}\nabla f_i(\boldsymbol{\theta}_{i,k}^e)\right)-\left(\sum_{i=1}^{n}\lambda_i\bar{E}_l^e\nabla f_i(\boldsymbol{\theta}^e)\right)\right\|^2\right]$$

$$\leq \underbrace{2\mathbb{E}\Big[\Big\|\sum_{i=1}^{n}\Delta_i^e\sum_{k=0}^{E_i-1}\nabla f_i(\boldsymbol{\theta}_{i,k}^e)\Big\|^2\Big]}_{\mathcal{C}}+\underbrace{2\mathbb{E}\Big[\Big\|\Big(\sum_{i=1}^{n}\lambda_i\sum_{k=0}^{E_i-1}\nabla f_i(\boldsymbol{\theta}_{i,k}^e)\Big)-\Big(\sum_{i=1}^{n}\lambda_i\bar{E}_l^e\nabla f_i(\boldsymbol{\theta}^e)\Big)\Big\|^2\Big]}_{\mathcal{D}}. \tag{B.5}$$

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

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

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:

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

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 (B.5), we get:

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

In the following, we first simplify Equation (B.4), and then, we plug the bounds above on $\mathcal{A}$ and $\mathcal{B}$ in it. We have:

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

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:

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

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

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

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

Lemma 8 (Bounded local drifts). *Suppose Assumption 1 holds. The local drift happening at client i during communication round e is bounded:*

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

where `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:

$$\begin{aligned}\mathbb{E}\|\boldsymbol{\theta}_{i,k}^e - \boldsymbol{\theta}^e\|^2 &= \mathbb{E}[\|\boldsymbol{\theta}_{i,k-1}^e - \eta_l \tilde{g}_i(\boldsymbol{\theta}_{i,k-1}^e) - \boldsymbol{\theta}^e\|^2] \\ &\leq \mathbb{E}[\|\boldsymbol{\theta}_{i,k-1}^e - \eta_l \nabla f_i(\boldsymbol{\theta}_{i,k-1}^e) - \boldsymbol{\theta}^e\|^2] + \eta_l^2 \sigma_{i,\tilde{g}}^2 \end{aligned} \tag{B.7}$$

where the inequality comes from Lemma 7. The first term on the right side of the above inequality can be bounded as:

$$\mathbb{E}[\|\boldsymbol{\theta}_{i,k-1}^e - \eta_l \nabla f_i(\boldsymbol{\theta}_{i,k-1}^e) - \boldsymbol{\theta}^e\|^2] \leq \left(1 + \frac{1}{2E_i - 1}\right) \mathbb{E}[\|\boldsymbol{\theta}_{i,k-1}^e - \boldsymbol{\theta}^e\|^2] + 2E_i \eta_l^2 \mathbb{E}\|[\nabla f_i(\boldsymbol{\theta}_{i,k-1}^e)\|^2], \tag{B.8}$$

where we have used Lemma 6. Now, we bound the last term in the above inequality. We have:

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

By using relaxed triangle inequality (Lemma 6) and Assumption 1, we get:

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

Now, we can rewrite Equation (B.8) and then Equation (B.7):

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

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

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

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:

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

Therefore, we have:

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

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 (B.6) and get:

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

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

Hence, we have:

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

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

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

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

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

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

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

which completes the proof. □

E Detailed results

E.1 Test accuracy comparison

In Tables B.7 to B.10, 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 B.7: Comparison of different algorithms (on MNIST, $E = 200$). FedAvg achieves 98.6%.

alg \ distr	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 B.8: Comparison of different algorithms (on FMNIST, $E = 200$). FedAvg achieves 90.28%.

alg \ distr	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**

E.2 Ablation study on privacy level and number of clients

The results in Tables B.11 to B.12 show the detailed results for the ablation study on privacy level and number of clients, reported in Figure 3.6 (left and middle figures, respectively). The values are the mean and standard deviation of average test accuracy across clients over three different runs.

Table B.9: Comparison of different algorithms (on CIFAR10, $E = 200$). FedAvg achieves 73.55%.

alg \ distr	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**

Table B.10: Comparison of different algorithms (on CIFAR100, $E = 200$). FedAvg achieves 61.80%.

alg \ distr	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 B.11: Detailed results for ablation study on the privacy level of clients in Figure 3.6, left.

alg \ distr	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	$\mathbf{90.71}_{\pm 0.65}$	$\mathbf{89.38}_{\pm 0.76}$	$\mathbf{85.13}_{\pm 0.68}$	$\mathbf{81.19}_{\pm 0.97}$

E.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 E.1 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 3.3. We can clearly observe that most of the eigen

Table B.12: Detailed results for ablation study on number of clients in Figure 3.6, middle.

alg \ distr	$n = 20$	$n = 40$	$n = 60$
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	$\mathbf{85.13}_{\pm 0.68}$	$\mathbf{76.85}_{\pm 0.75}$	$\mathbf{72.77}_{\pm 0.78}$

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 E.2, 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 E.2. We have done similar comparisons in the next section (Appendix F) for other heterogeneity scenarios.

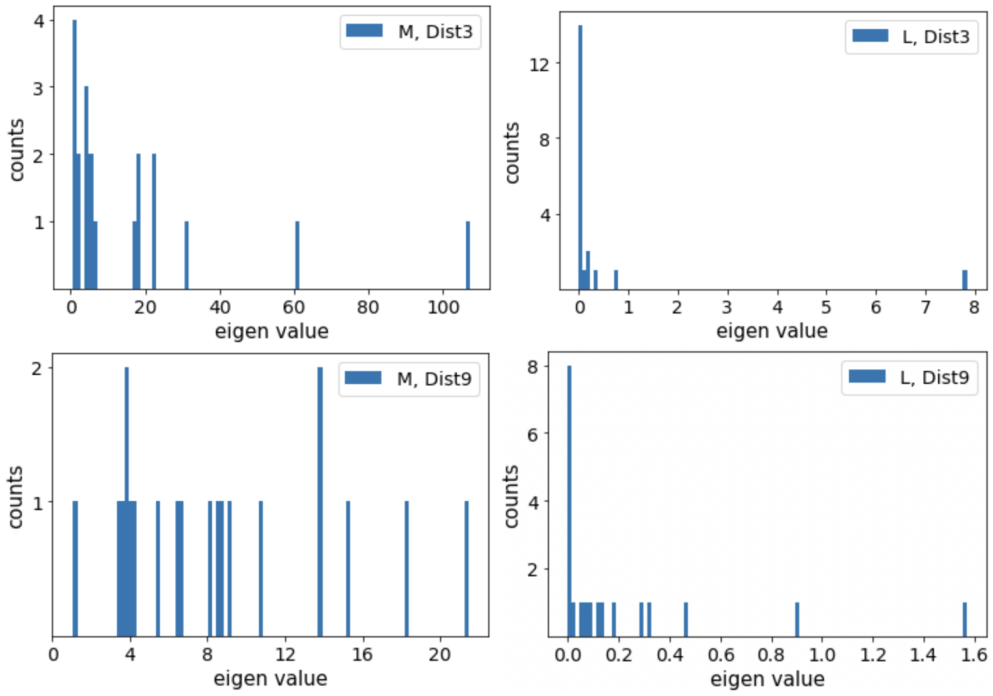


Figure E.1: Comparison of the eigen values of matrices $\mathbf{M}$ (left) and $\mathbf{L}$ (right) on MNIST dataset. The concentration of eigen values in the right figures around 0 shows that the matrix $\mathbf{L}$ returned by Robust PCA, is indeed low rank, while $\mathbf{M}$ is not, due to the noise existing in clients model updates. The results for these experiments were reported in Table B.7 (Robust-HDP, Dist3 and Dist9).

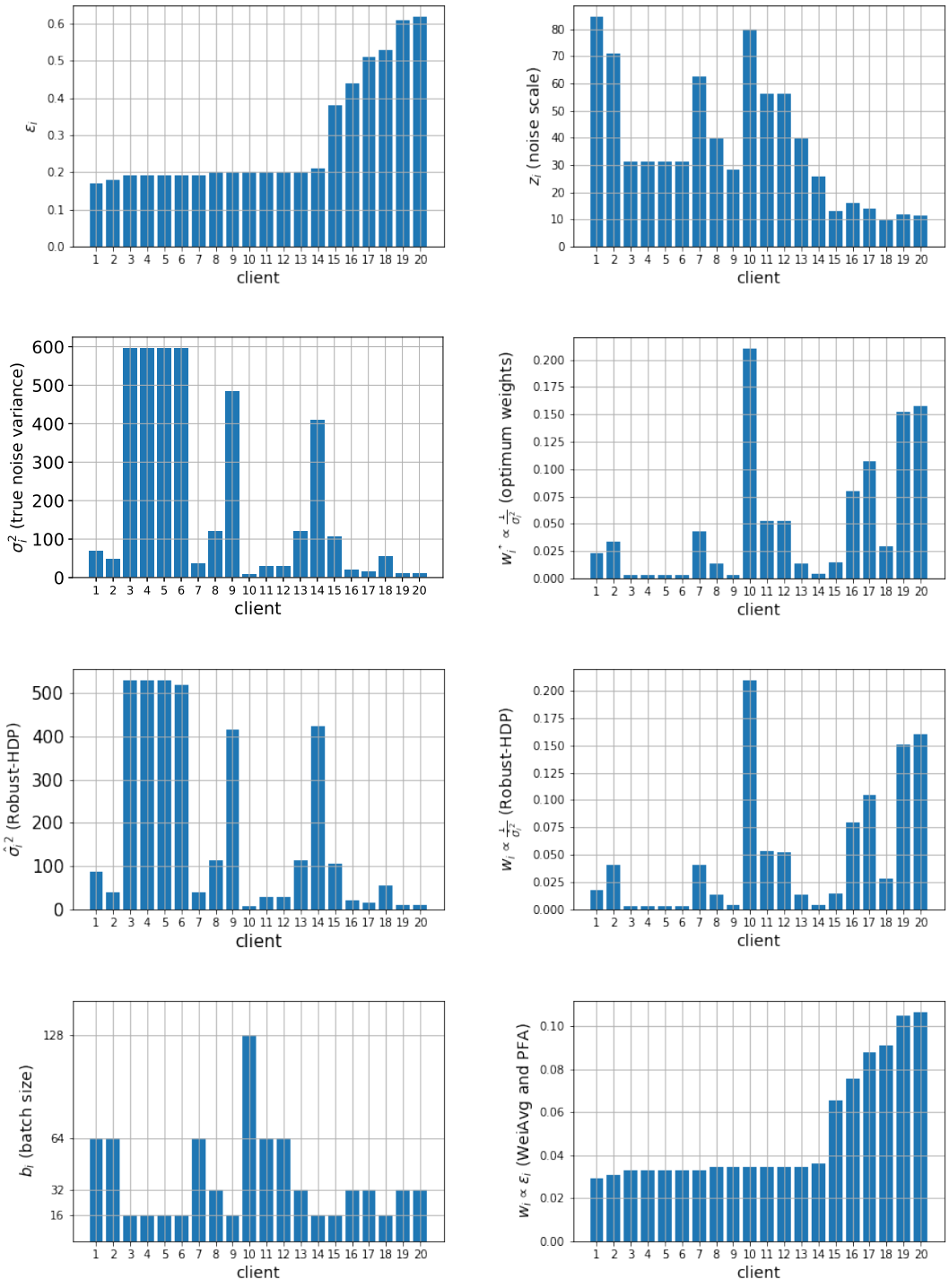


Figure E.2: Comparison of weight assignments for Dist8 and MNIST with the data split in Table B.2. The assigned weights by baseline algorithms show that their weight assignment strategies are not based on the noise variance in clients model updates, hence suboptimal. The results for this experiment were reported in Table B.7 (Robust-HDP, Dist8).

F 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. 2020) 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 $\mathrm{Dir}(1)$. 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.

F.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 Section F.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 3.3. In this case, our analysis in Section 3.5 can be rewritten as follows (as before, we use the same $\delta_i = \delta$ and $K_i = K$ for all clients):

Table B.13: 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.

alg \ distr	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

1. Effective clipping threshold: when the clipping is indeed effective for all samples, the variance of the noisy stochastic gradient in Equation (3.1) can be computed as:

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

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

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

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

Finally:

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

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 3.3, 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 B.13 on CIFAR10, which shows superiority of Robust-HDP in this heterogeneity scenario.

F.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:

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

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

2. Ineffective clipping threshold:

$$\mathbb{E}[\tilde{g}_i(\boldsymbol{\theta})] = \mathbb{E}[g_i(\boldsymbol{\theta})] = \nabla f_i(\boldsymbol{\theta}),$$
$$\sigma^2_{i,\tilde{g}} = \texttt{Var}(\tilde{g}_i(\boldsymbol{\theta})) = \texttt{Var}(g_i(\boldsymbol{\theta})) + \frac{p\sigma^2_{i,\texttt{DP}}}{b^2} \leq \sigma^2_{i,g} + \frac{pc^2z^2(\epsilon,\delta,\frac{b_i}{N_i},K,E)}{b_i^2},$$

and

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

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 B.14, 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 3.3 for ϵ, i.e. $\epsilon \in \{2.6, 2.0, 1.1, 0.6, 0.35\}$. Also, as before, we have fixd δ_i to $1\texttt{e}-4$.

F.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:

Table B.14: 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**

1. Effective clipping threshold:

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

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

2. Ineffective clipping threshold:

$$\mathbb{E}[\tilde{g}_i(\boldsymbol{\theta})] = \mathbb{E}[g_i(\boldsymbol{\theta})] = \nabla f_i(\boldsymbol{\theta}),$$
$$\sigma^2_{i,\tilde{g}} = \texttt{Var}(\tilde{g}_i(\boldsymbol{\theta})) = \texttt{Var}(g_i(\boldsymbol{\theta})) + \frac{p\sigma^2_{i,\texttt{DP}}}{b^2} \leq \sigma^2_{i,g} + \frac{pc^2z^2(\epsilon,\delta,\frac{b}{N_i},K,E)}{b^2},$$

and

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

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 (3.6) 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.

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:

$$w_i^* \propto \frac{1}{\sigma_i^2} \approx \frac{b}{K\eta_l^2} \cdot \frac{1}{N_i \sigma_{i,\tilde{g}}^2} \approx \frac{b^3}{Kpc^2\eta_l^2} \cdot \frac{1}{N_i z^2(\epsilon, \delta, \frac{b}{N_i}, K, E)} = \frac{b^3}{Kpc^2\eta_l^2} \cdot \frac{1}{H(N_i, b, \epsilon, \delta, K, E)} \tag{B.9}$$

where we have used Equation (3.3) (with b and ϵ) and $H(N_i, b, \epsilon, \delta, K, E) := N_i z^2(\epsilon, \delta, \frac{b}{N_i}, K, E)$. Now note that z decreases with N_i sublinearly (see Figure 3.2. Remember that $q_i = \frac{b_i}{N_i}$). We have plotted the behavior of the function $H(N_i, b, \epsilon, \delta, K, E)$ as a function of N_i in Figure F.3. 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 Equation (3.6)) 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.

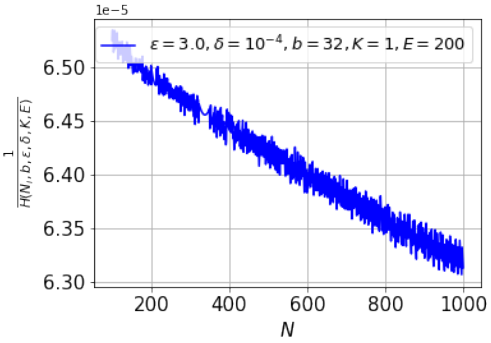


Figure F.3: The behavior of function $1/H(N, b, \epsilon, \delta, K, E)$ as a function of N (dataset size) for an instance value of $\epsilon = 3$ and batch size $b = 32$. Oscillations in the curves are due to finding z empirically.

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 and 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 (B.9) and Figure F.3. 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

Table B.15: 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.

alg \ distr	$\epsilon = 2.6$	$\epsilon = 2.0$	$\epsilon = 1.1$	$\epsilon = 0.6$	$\epsilon = 0.35$
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**

Table B.16: 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.

alg \ distr	$\epsilon = 2.6$	$\epsilon = 2.0$	$\epsilon = 1.1$	$\epsilon = 0.6$	$\epsilon = 0.35$
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**)

Dist9 from Table 3.3 for ϵ, i.e. $\epsilon \in \{2.6, 2.0, 1.1, 0.6, 0.35\}$. Also, as before, we fix δ_i to $1e-4$. With these settings and using the Dirichlet data allocation mentioned before, we got the results in Tables B.15 and B.16.

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

G 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 Section 3.5.2 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 the following.

H.1 Robust-HDP with moderately heterogeneous data distribution

In order to evaluate Robust-HDP under a moderately heterogeneous data split, we run experiments on MNIST. In order to simulate a controlled higher data heterogeneity, we use the sharding data splitting method described in Appendix A.1 and Table B.2, *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 3.5. As we observed, Robust-HDP still outperforms the baselines in most of the cases. However, compared to the results in Table B.7, 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 G.4. Remember that, we have assumed uniform batch size of 128 for all clients. Therefore, the only factor that makes variation in $\{\sigma_i^2\}$ is the clients' privacy parameters $\{\epsilon_i\}$ being different. There are multiple points in Figure G.4: 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 G.4 with that in Figure E.2, 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 3.5.6, 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.

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 in 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 3.5, 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 $\{16, 32, 64, 128\}$: we recall Equation (3.5), 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 B.17 are obtained in this case. Also, Figure G.5 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 G.5. First, the accuracy of RPCA decomposition in estimating $\{\sigma_i^2\}$ has again decreased compared to that in Figure E.2, 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 G.4: 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 G.4). 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

Table B.17: 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.

alg \ distr	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**

batch size heterogeneity simultaneously, and the results in Table B.17 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 Section F.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 3.5, 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.

H.2 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 highly heterogeneous data splits. In non-private FL systems, high data heterogeneity is usually addressed by personalized FL (Li et al. 2020b) 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 splits, clustered FL is more common (Sattler et al. 2019; Werner et al. 2023).

In the existence of a trusted server, 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 with $\delta = 0$). When there is no trusted server, 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. We further explore this setting in Chapter 4.

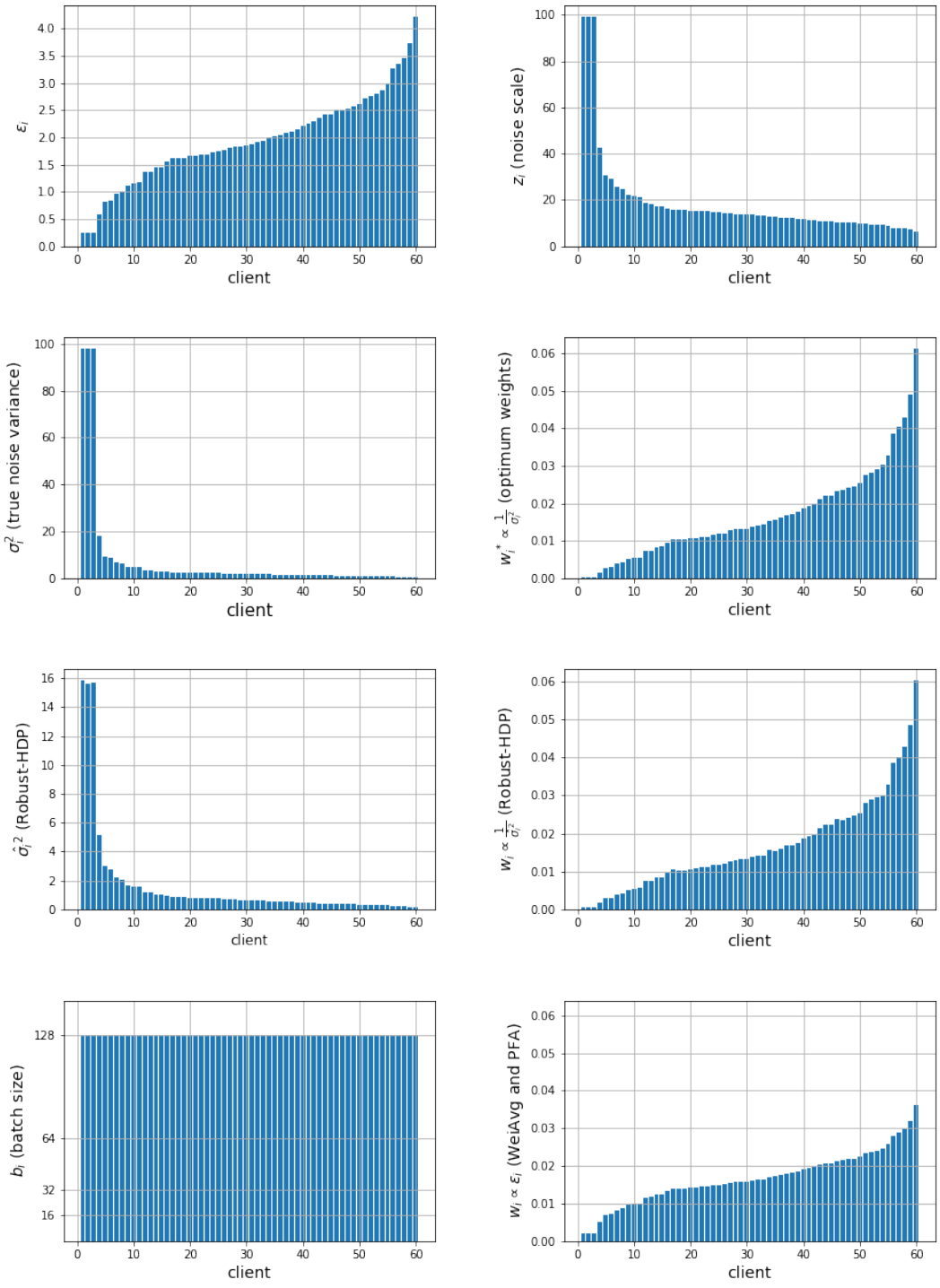


Figure H.4: Comparison of weight assignments by different algorithms on MNIST, Dist1 and 60 clients with heterogeneous data distribution (maximum 8 labels per client) and uniform batch size ($b_i = 128$). The first 40 (last 20) clients, which have the smallest (largest) ϵ privacy parameters, get assigned smaller (larger) weights by Robust-HDP than by PFA and WeiAvg. The results for this experiment were reported in Table 3.5 (Robust-HDP, Dist1).

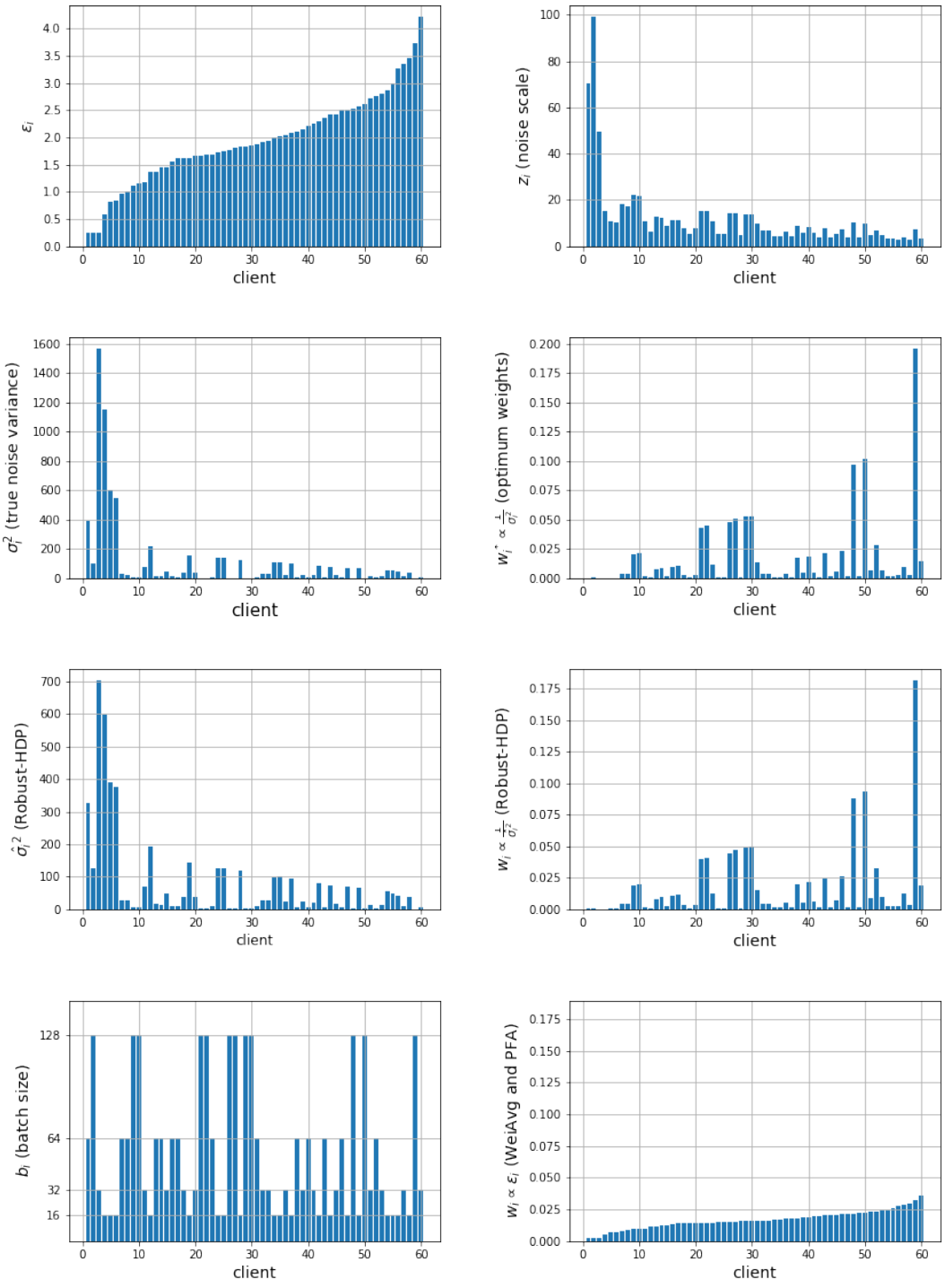


Figure H.5: Comparison of weight assignments by different algorithms on MNIST, Dist1 and 60 clients with hetrogeneous data distribution (maximum 8 labels per client) and heterogeneous batch sizes. The weight assignments by Robust-HDP are close to the optimal weight assignment strategy, despite the heterogeneity in the data split. The results for this experiment were reported in Table B.17 (Robust-HDP, Dist1).

Appendix C

Appendix of Chapter 4

A Privacy accounting and budgeting

In the following, we add some useful results to the ones introduced in Chapter 1 and explain about our method for privacy budgeting and accounting, with further details deferred to the next sections.

According to Theorem 1, RDP can be used for composition of *heterogeneous* private mechanisms, e.g., for accounting privacy of R-DPCFL, which uses different batch sizes in the first and the subsequent rounds. We also use Lemma 3 for conversion of (α, ϵ)-RDP to standard (ϵ, δ)-DP. The Gaussian mechanism satisfies (α, ϵ)-RDP, based on the following Proposition from (Mironov 2017):

Proposition 1. *If $f : \mathcal{A} \to \mathcal{R}$ has sensitivity c, then its randomization with a Gaussian mechanism with noise variance σ^2_{DP} satisfies $(\alpha, \frac{\alpha c^2}{2\sigma^2_{DP}})$-RDP.*

In order to have a fair comparison between R-DPCFL and the baselines, we align them all to have the same "total" privacy budget ϵ and satisfy (ϵ, δ)-DP for a fixed δ. In order to account the privacy of an algorithm, we compose the RDP guarantees of all private operations in the algorithm and then convert the resulting RDP guarantee to approximate (ϵ, δ)-DP by using Lemma 3. The DPSGD performed by different algorithms for local training benefits from privacy amplification by subsampling (Mironov et al. 2019). Algorithms that have privacy overheads, e.g., IFCA and R-DPCFL which need to privatize their local clustering as well, will have less privacy budget left for training. In other words, for the same total privacy budget ϵ, IFCA and R-DPCFL will use a larger amount of noise when running DPSGD, compared to MR-MTL that has zero privacy overhead.

B Experimental setup

B.1 Datasets

Data split: We use three datasets MNIST, FMNIST and CIFAR10, and consider a distributed setting with 21 clients. In order to create majority and minority clusters, we consider 4 clusters with different number of clients $\{3, 6, 6, 6\}$ (21 clients in total). The first cluster with the minimum number of clients is the "minority" cluster, and the last three are the "majority" ones. The data distribution $P(x, y)$ varies across clusters. We use two methods for making such data heterogeneity: 1. **covariate shift** 2. **concept shift**. In covariate shift, we assume that features marginal distribution $P(x)$ differs from one cluster to another cluster. In order to create this variation, we first allocate samples to all clients in an *uniform* way. Then we rotate the data points (images) belonging to the clients in cluster k by $k * 90$ degrees. For concept shift, we assume that conditional distribution $P(y|x)$ differs from one cluster to another cluster, and we first allocate data samples to clients in a uniform way, and flip the labels of the points allocated to clients: we flip y_{ij} (label of the j-th data point of client i, which belongs to cluster k) to $(y_{ij} + k)$ *mod* 10, The local datasets are balanced–all users have the same amount of training samples. The local data is split into train and test sets with ratios 80%, and 20%, respectively. In the reported experimental results, all users participate in each communication round.

Table C.1: 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		

B.2 Models and optimization

We use a simple 2-layer CNN model with ReLU activation, the detail of which can be found in Table B.1 for MNIST and FMNIST. Also, we use the residual neural network (ResNet-18) defined in (He et al. 2015), which is a large model. To update the local models allocated to each client during each round, we apply `DPSGD` (Abadi et al. 2016) with a noise scale z which depends on some parameters, as in Lemma 4.

In order to simulate an `FL` setting, where clients (silos) have large local datasets and there is a structured data heterogeneity across clusters, we split the full dataset between the clients belonging

Table C.2: Details of the used datasets in the main body of the paper. ResNet-18 is the residual neural networks defined in He et al. 2015. CNN: Convolutional Neural Network defined in Table B.1.

Datasets	Train set size	Test set size	Data Partition method	# of clients	Model	# of parameters
MNIST	48000	12000	covariate shift	$\{3, 6, 6, 6\}$	CNN	28,938
FMNIST	48000	12000	covariate shift	$\{3, 6, 6, 6\}$	CNN	28,938
CIFAR10	40000	10000	covariate and concept shift	$\{3, 6, 6, 6\}$	ResNet-18	11,181,642

to each cluster. This way, each client gets $8,000$ train and $1,666$ test samples for MNIST and FMNIST. Also, each client gets $10,000$ and $1,666$ train and test samples for CIFAR10 dataset (both covariate shift and concept shift).

B.3 MR-MTL formulation

The objective function of Mean-Regularzied Multi-Task Learning (MR-MTL) can be expressed as:

$$\min_{\boldsymbol{\theta}_i, i\in\{1,\cdots,n\}} \sum_{i=1}^{n} g_i(\boldsymbol{\theta}_i) \quad \text{with} \quad g_i(\theta_i) = f_i(\boldsymbol{\theta}_i) + \frac{\lambda}{2}\|\boldsymbol{\theta}_i - \bar{\boldsymbol{\theta}}\|_2^2,$$

where $\bar{\boldsymbol{\theta}} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{\theta}_i$ is the average model parameter across clients and $f_i(\boldsymbol{\theta}_i)$ is the loss function of personalized model parameter θ_i of client i on its local dataset $\mathcal{D}_i$. With $\lambda = 0$, MR-MTL reduces to local training. A larger regularization term λ encourages local models to be closer to each other. However, MR-MTL may not recover FedAvg (McMahan et al. 2017a) as $\lambda \to \infty$. See algorithm A1 in (Liu et al. 2022) for more details about MR-MTL.

B.4 Tuning hyperparameters of baseline algorithms

Section 4.6.1 explains our criteria for baseline selection. We compare our R-DPCFL algorithm, which benefits from robust clustering, with four baseline algorithms, including: 1) DPFedAvg (Noble et al. 2021), which learns one global model for all clients, and is called Global in the paper 2) Local, in which clients do not participate FL and run DPSGD locally to train a model solely on their local dataset 3) MR-MTL personalized FL algorithm (Liu et al. 2022), which learns a global model and one personalized model for each client 4) A DP extension of the clustered FL algorithm IFCA (Ghosh et al. 2020) to DPFL systems enhanced with exponential mechanism (see Section B.5) 5) An oracle algorithm, which has the knowledge of the true underlying clients' clusters, which we call O-DPCFL.

For all algorithms and all datasets, we set total number of rounds E to 200 and per-round number of local epochs K to 1. Following (Abadi et al. 2016), we set the batch size of each client

such that the number of batches per epoch is in the same order as the total number of epochs: $N_i/b_i^e = E \cdot K = 200$. For MNIST and FMNIST, this leads to batch sizes $b_i^e = 32$ for all clients i and every round e for the baseline algorithms. For CIFAR10 (covariate shift and concept shift), this leads to batch size $b_i^e = 64$ for all clients i and every round e for the baseline algorithms. While R-DPCFL uses full batch sizes in the first round (i.e. $b_i^1 = N_i$ for all i), it needs to use small batch sizes in the next rounds. We have further explained about this in Section C.1.

Having determined the batch size for all algorithms, clipping threshold c and learning rate η_l are determined via a grid search on clients' validation sets. For each algorithm and each dataset, we find the best learning rate from a grid: the one which results in the highest average accuracy at the end of `FL` training on a "validation set" with size 1666 samples for each client. We use the grid $\eta_l \in$ `{5e-4, 1e-3, 2e-3, 5e-3, 1e-2, 2e-2, 5e-2, 1e-1}` for all datasets and all algorithms. Similarly, we use the grid $c \in$ `{1, 2, 3, 4, 5}` for setting the clipping threshold for all datasets and all algorithms based on the clients' validation sets.

B.5 Implementation of private local clustering for IFCA and R-DPCFL

In every round of IFCA and during the rounds $e > E_c$ of R-DPCFL, the server sends M cluster models to all clients, and they evaluate them on their local datasets. Then, each client i selects the model with the lowest loss on its local dataset $\mathcal{D}_i$, trains it for K local epochs and sends the result back to the server. This model selection performed by each client can lead to privacy leakage w.r.t its local dataset, if it is not privatized. In order to protect data privacy, clients need to privatize their local clustering by using exponential mechanism and accounting its privacy using `zCDP`, explained in Section 1.2.5. Assuming a total privacy budget ϵ for a client i, it has to split the budget between private clustering and `DPSGD`. Naive split of privacy budget can lead to very noisy `DPSGD` steps or very noisy local selection by clients. Following Liu et al. 2022, we use two strategies to mitigate the privacy overhead of local clustering performed by IFCA and R-DPCFL:

- **Clients use model accuracy, instead of loss, as the score function for model selection:** clients use model accuracy as score function $s(\mathcal{D}_i, m)$ evaluating cluster model m on client i's dataset. The reason is that while, loss function has practically an unbounded sensitivity to individual samples in the clients' datasets, model accuracy is a low-sensitivity function, espcially in cross-silo `FL` settings with large local datasets. More specifically, let us assume client i with local dataset $\mathcal{D}_i$ (which has size N_i) uses models' accuracy on $\mathcal{D}_i$ for model selection. It can be shown that under all add/remove/replace notions of dataset neighborhood, sensitivity of model accuracy (as score function) is bounded as follows (Liu et al. 2022):

$$\Delta_{acc} = \max_{m \in [M]} \max_{\mathcal{D}_i, \mathcal{D}_i'} |s(\mathcal{D}_i, m) - s(\mathcal{D}_i, m)| \leq \frac{1}{N_i - 1}.$$

  Since local dataset sizes are usually large, especially in cross-silo `FL`, the sensitivity of model accuracy is much smaller than that of model loss. Therefore, following Liu et al.

2022, we set the per-round privacy budget of private model selection to a very smalle value $\epsilon_{\text{select}} = 0.03 \cdot \epsilon$ (3% of the total privacy budget). Yet, the cost of private selection by clients can grow quickly if clients naively run local clustering for "many" rounds. Therefore, we use the following strategy as well. It is noteworthy that in our experiments, we observed that IFCA baseline algorithm performs better when clients use model train accuracy (instead of train loss) for cluster selection.

- **Reduce the number of rounds with local clustering on clients' side:** Clients run local clustering for less rounds. Following Liu et al. 2022, we let clients run local clustering for the 10% of the total number of rounds E. For example, IFCA runs local clustering during the first $\lfloor \frac{E}{10} \rfloor$ rounds, and fixes clients' cluster assignments afterwards. Similarly, R-DPCFL lets clients run local clustering during rounds $E_c \leq e \leq E_c + \lfloor \frac{E}{10} \rfloor$, and fixes clients' cluster assignments afterwards.

The privacy overhead of private model selection can still grow and leave a low privacy budget for training with DPSGD. Choosing a small selection budget ϵ_{select} leaves most of the total privacy budget ϵ for training with DPSGD, but leads to noisy and inaccurate cluster selection by clients. Similarly, a large ϵ_{select} leads to more noisy gradient steps by DPSGD.

B.6 DP privacy parameters

For each dataset, 5 different values of ϵ (the total privacy budget) from set $\{3, 4, 5, 10, 15\}$ are used. We fix δ for all experiments to 10^{-4}, which satisfies $\delta < N_i^{-1}$ for every client i. We use the Rényi DP (RDP) privacy accountant (TensorFlow privacy implementation (McMahan et al. 2019)) during the training time. This accountant is able to handle the difference in the batch size of R-DPCFL between the first round $e = 1$ and the next rounds $e > 1$ by accounting the composition of the corresponding *heterogeneous* private mechanisms.

B.7 Gaussian mixture model

We use the Gaussian Mixture Model of Scikitlearn, which can be found here: https://scikit-learn.org/dev/modules/generated/sklearn.mixture.GaussianMixture.html. The GMM model has three hyper-parameters:

1) parameter initialization, which we set to "k-means++". This is because this type of initialization leads to both low time to initialize and low number of EM iterations for the GMM to converge (Arthur and Vassilvitskii 2007; Biernacki et al. 2003).

2) Type of the covariance matrix, which we set to "spherical", i.e. each component has a diagonal covariance matrix with a single value as its diagonal elements. This is in accordance with Equation (C.6) and that we know the covariance matrices should be diagonal.

3) Finally, the number of components (clusters) is either known or it is unknown. In the latter case, we have explained in Section C.3 how we can find the true number of clusters by using the confidence level (MSS) of the GMM model.

C Setting hyper-parameters of R-DPCFL

As explained in the paper, R-DPCFL has three hyperparameters, which we explain how to set in the following:

C.1 Batch size $b_i^{>1}$

Batch size $b_i^{>1}$, which is the batch size used by R-DPCFL during the rounds $e > 1$, has to be set to a small value, as observed in Figure 4.3 right. R-DPCFL is not sensitive to this parameter, as long as a small value is chosen for it. For the results in the paper, we use $b_i^{>1} = 32$ for all experiments with R-DPCFL. We further explain about the effect of this parameter as follows:

As we observed in Lemma 4 and Figure 4.3 left, $\texttt{Var}(\Delta\tilde{\boldsymbol{\theta}}_i^1(b_i^1)|\boldsymbol{\theta}^{init})$ is an increasing function of $b_i^{>1}$. More generally, the effect of increasing $b_i^{>1}$ is on three things: 1) increasing noise variance $\texttt{Var}(\Delta\tilde{\boldsymbol{\theta}}_i^1(b_i^1)|\boldsymbol{\theta}^{init})$ (as shown in Figure 4.3, left) 2) decreasing noise variance $\texttt{Var}(\Delta\tilde{\boldsymbol{\theta}}_i^1(b_i^e)|\boldsymbol{\theta}_i^{e,0})$ (as shown in Figure 4.3, right) 3) decreasing number of gradient steps during each round e for $e > 1$. While the first one is only limited to the first round $e = 1$, the last two affect the remaining $E - 1$ rounds and have conflicting effects on the final accuracy. However, an important point about the problem of DP clustered FL is that finding the true structure of clusters in the first round is a prerequisite for making progress in the next rounds. Therefore, increment in noise variance $\texttt{Var}(\Delta\tilde{\boldsymbol{\theta}}_i^1(b_i^1)|\boldsymbol{\theta}^{init})$ (the first effect) is the most important one. We have demonstrated this effect in Figure C.1, which shows that how increasing $b_i^{>1}$ adversely affects the clustering done at the end of the first round. Note how MSS score of the learned GMM increases as $b_i^{>1}$ increases. Therefore, in order to have a reliable client clustering at the end of the first round, we need to keep the value of $b_i^{>1}$ to small values: the smaller total privacy budget ϵ, the smaller value should be used for $b_i^{>1}$. Following this observation, we have fixed $b_i^{>1}$ to 32 in all our experiments with R-DPCFL.

C.2 The strategy switching time E_c

The strategy switching time E_c can also be set by using the uncertainty metric $\texttt{MPO} \in [0, 1)$. Intuitively, if the learned GMM is not certain about its clustering decisions, R-DPCFL should not rely on its decisions for a large E_c, and vice versa. Hence, we can set E_c as a decreasing function of MPO. For instance, $E_c = (1 - \texttt{MPO})\frac{E}{2}$, which is used in this work, means that if a GMM is completely confident about its clusterings, e.g., what happens in Figure 4.5 right, the server changes the clustering strategy to loss-based after the first half of rounds. As the uncertainty increases, this change happens earlier (e.g., when ϵ is small), and R-DPCFL slowly gets close to the existing loss-based clustering methods like IFCA (Ghosh et al. 2020).

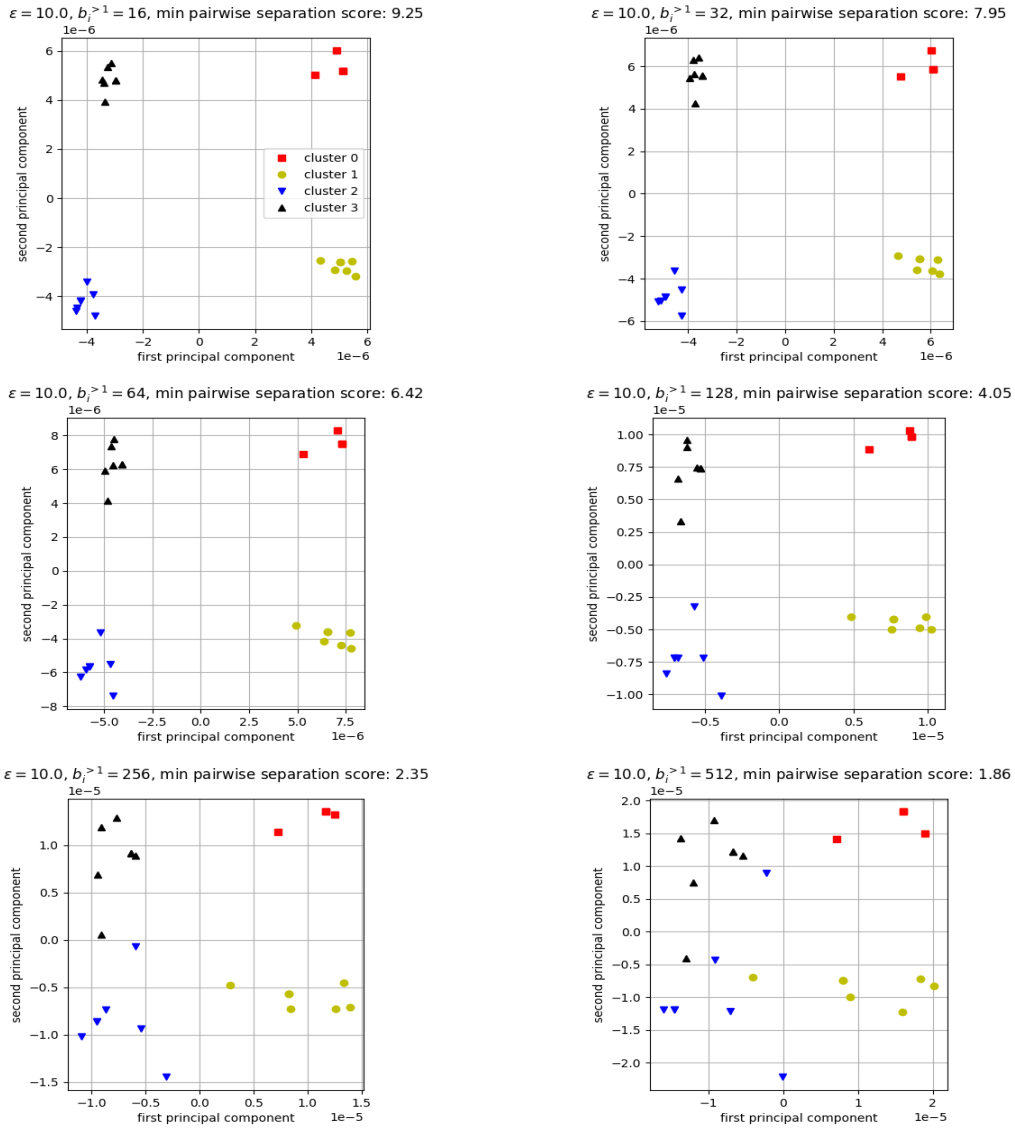


Figure C.1: The effect of increasing the batch size after the first round, i.e. $b_i^{>1}$, on the model updates $\{\Delta\tilde{\boldsymbol{\theta}}_i^1\}_{i=1}^n$ at the end of the first round. All clients have used full batch sizes in the first round, i.e. $\forall i : b_i^1 = N_i$. The level of noise in $\{\Delta\tilde{\boldsymbol{\theta}}_i^1\}_{i=1}^n$ affects the quality and confidence of the client clustering that the server performs at the end of the first round. As can be observed, for a fixed $\epsilon = 10$, the model updates scatter further in space as $b_i^{>1}$ increases and different clusters get less separated. This leads to a decrement in the confidence level or the (MSS) score of the resulting GMM, as $b_i^{>1}$ increases (mentioned on the top of each plot). All the results are obtained on CIFAR10 with covariate shift (rotation) across clusters.

C.3 The number of clusters M

Knowing the number of clusters is broadly accepted and applied in the clustered FL literature (Briggs et al. 2020; Ghosh et al. 2020; Ruan and Joe-Wong 2021). **This is the assumption of our baseline algorithms too**. Yet, techniques to determine the number of clusters can enable our approach to be more widely adopted. In this section, we show that how we can find the true number of clusters (M) when it is not given. Our method relies on the `MSS` score (confidence level) defined in Section 4.5.5: $\texttt{MSS} = \min_{m,m'} \hat{\texttt{SS}}(m, m') \in [0, +\infty)$ (please see the detailed explanations in Section 4.5.5). Consider the Figure 4.5 right as an example. There is a good separation between the $M = 4$ existing clusters, thanks to clients using full batch sizes in the first round. Fitting a `GMM` with 4 components to the model updates results in the highest `MSS` for the learned `GMM` model: remember that `MSS` was the maximum pairwise separation score between the different components of the learned `GMM`. In contrast, if we fit a `GMM` with 3 components (less than the true number of components) to the same model updates in the figure, then two clusters will be merged into one component (for examples clusters 0 and 1) leading to a high radius for one of the three components of the resulting `GMM`. This leads to a low `MSS` (confidence level) for the resulting `GMM`. Similarly, if we fit a `GMM` with 5 components, one of the four clusters (for example cluster 1) will be split between two of the 5 components (call them m and m'), which leads to a low inter-component distance ($\Delta_{m,m'}$) for the pair of components. This also leads to a low `MSS` for the resulting `GMM`. However, **fitting a `GMM` with $M = 4$ components leads to a well separation between all the true components and maximizes the resulting `MSS`**. Based on this very intuitive observation, we propose the following method for setting m at the end of the first round: We select the number of clusters/components, which leads to the maximum `MSS` for the resulting `GMM`. More specifically:

$$M = \underset{m \in S}{\operatorname{argmax}}\, \texttt{MSS}\Big(\texttt{GMM}(\Delta\tilde{\boldsymbol{\theta}}_1^1, \ldots, \Delta\tilde{\boldsymbol{\theta}}_n^1; m)\Big), \text{(line 10 of Algorithm 2)}$$

where S is a set of candidate values for M: at the end of the first round and on the server, we learn one `GMM` for each candidate value in S on the same received model updates $\{\Delta\tilde{\boldsymbol{\theta}}_i^1\}_{i=1}^n$. Finally, we choose the value resulting in the `GMM` with the highest `MSS` (confidence). Therefore, this method is run on the server and does not incur any additional privacy overheads. It is also noteworthy that we know from Lemma 5 that learning the `GMM` does not incur much computational cost when large enough (and small enough) batch sizes are used in the first round (subsequent rounds).

We have evaluated this method on multiple data splits and different privacy budgets (ϵ) on CIFAR10, MNIST and FMNIST. The method could predict the number of underlying clusters with 100% accuracy for the MNIST and FMNIST datasets for all values of ϵ. Results for CIFAR10 are shown in Figure C.2. As can be observed, the method has made only one mistake for $\epsilon = 4$ (seed 1) and two mistakes for $\epsilon = 3$ (seeds 0 and 1), out of 20 total experiments. Even in those three cases, it has predicted M as 5, which is closest to the true value ($M = 4$) and does not lead to much performance drop (because having $M = 5$ splits an existing cluster into two and it is better

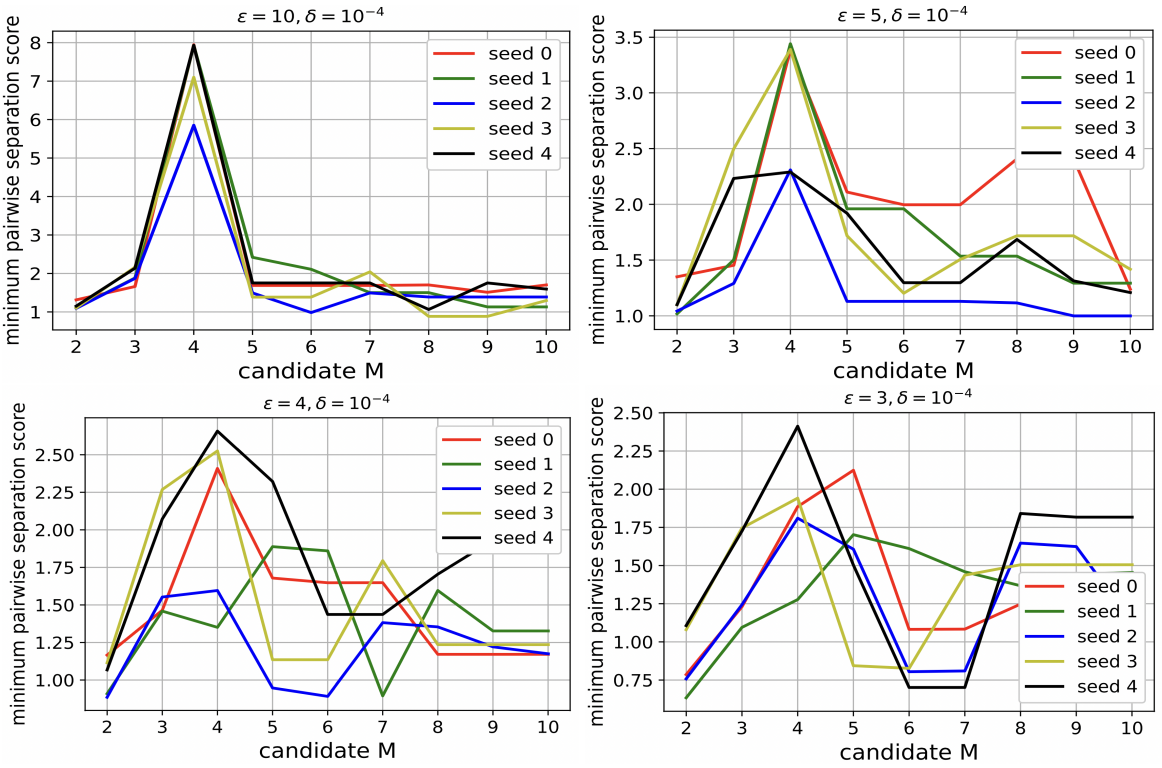


Figure C.2: The minimum pairwise separation score (MSS) or confidence of the GMM learned on $\{\Delta\tilde{\boldsymbol{\theta}}_i^1\}_{i=1}^n$ peaks at the true cluster number, which is equal to 4 in all the plots above. Each figure is for a different value of ϵ (mentioned on top of each figure), and are obtained on CIFAR10 with covariate shift (rotation) across clusters, and 5 different random data splits (5 seeds). All the results are obtained with full batch sizes in the first round and $b_i^{>1} = 32$ for all i. We can use this observation as a method to find the true number of clusters (M) when it is not given. For larger ϵ, this method work perfectly and even when ϵ is too small, e.g., $\epsilon = 3$, this method works well and predicts the true number of clusters correctly most of the times: 3 out of the 5 curves in the bottom right plot have a peak at $M = 4$ (the true cluster number). and the other 2 curves predict 5 as the true number, which is the closest and the best alternative for the true value $M = 4$.

than predicting for example $M = 3$, which results in "mixing" two clusters with heterogeneous data). Even in this cases, we can improve the prediction accuracy further by using smaller values of $b_i^{>1}$ (simultaneously with full batch sizes $b_i^1 = N_i$), e.g., $b_i^{>1} = 16$ or $b_i^{>1} = 8$, instead of $b_i^{>1} = 32$ in the figure above. This improvement happens as reducing $b_i^{>1}$ constantly enhances the separation between the underlying components (See Figure C.1), which leads to higher prediction accuracy.

Finally, note that none of the existing baseline algorithms has such an easy and applicable strategy for finding M. This shows another useful feature of the proposed R-DPCFL, which makes it more applicable to DP clustered FL settings.

D Complete experimental results

Figures D3 and D.4, which include the results for the Global baseline, are the more complete versions of the figures in the paper (Figure 4.6 and Figure 4.7).

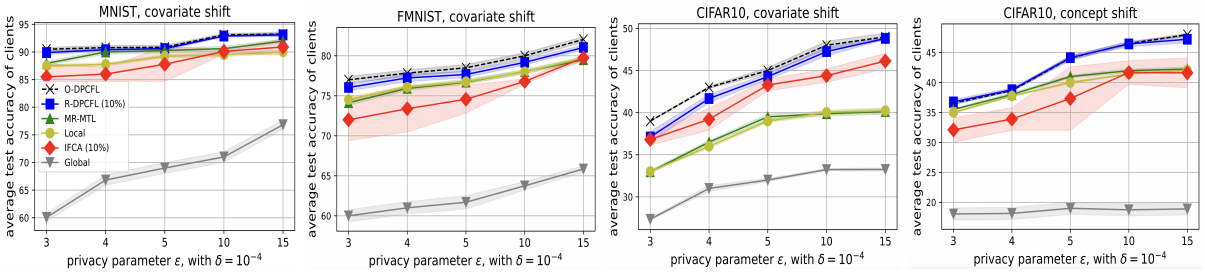


Figure D.3: Average test accuracy across clients for different total privacy budgets ϵ (results are obtained from 4 different random seeds). 10% means performing loss-based clustering by clients only in 10% of the total rounds (E).

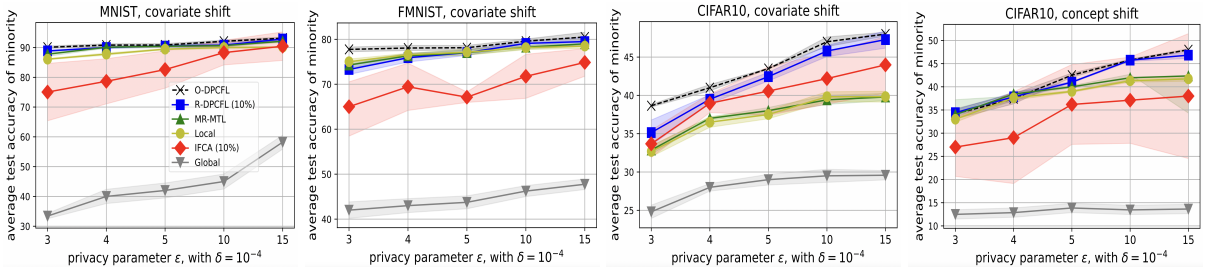


Figure D.4: Average test accuracy across clients belonging to the minority cluster for different total privacy budgets ϵ (results are obtained from 4 different random seeds). 10% means performing loss-based clustering by clients only in 10% of the total rounds (E).

Figure D.5 shows how the `MSS` score of the learned `GMM` model at the end of the first round can be indicative of whether the true clients' clusters will be detected or not. As observed, an `MSS` score above 2 always yields to correct detection of all clusters.

Finally, the results in Figure D.6 supports Theorem 11 by showing that if b_i^1 is large enough, the convergence rate of `EM` algorithm for learning the `GMM` at the end of the first round increases with b_i^1.

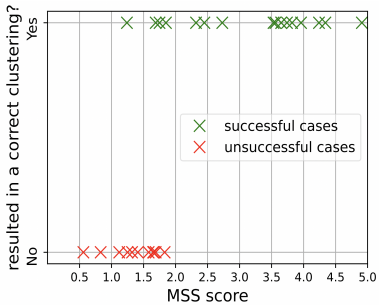


Figure D.5: The MSS score of the learned GMM is indicative of whether the true underlying clusters will be detected or not: an MSS score above 2 always leads to correct detection of clusters. Each point is the result of one independent experiment.

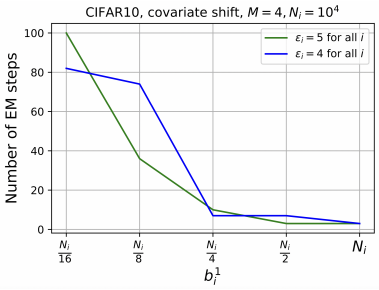


Figure D.6: The number of EM iterations needed for learning the GMM decreases as b_i^1 increases. Especially, at full batch size ($\forall i : b_i^1 = N_i$, very few iterations of EM are needed. The results are obtained on CIFAR10 with covariate shift across $M = 4$ clusters. According to Lemma 5, using a large enough batch size b_i^1 in the first round makes the underlying clusters in $\{\Delta\tilde{\boldsymbol{\theta}}_i^1\}_{i=1}^n$ more distinguishable. Consequently, according to Theorem 11, convergence rate of EM algorithm for learning the GMM increases with b_i^1.

E Proofs

E.1 Proof of Lemma 4

Lemma 4. *Let us assume $\boldsymbol{\theta}_i^{e,0}$ is the model parameter passed to client i at the beginning of round e. After K local epochs with step size η_l, the client generates the noisy DP model update $\Delta\tilde{\boldsymbol{\theta}}_i^e(b_i^e)$ at the end of the round. The amount of noise in the resulting model update can be quantified as:*

$$(\sigma_i^e(b_i^e))^2 := Var(\Delta\tilde{\boldsymbol{\theta}}_i^e(b_i^e)|\boldsymbol{\theta}_i^{e,0}) \approx K \cdot N_i \cdot \eta_l^2 \cdot \frac{pc^2 z_i^2(\epsilon, \delta, b_i^1, b_i^{>1}, N_i, K, E)}{(b_i^e)^3}.$$

Proof. The following proof has some common parts with the results in Section 3.5.1. We consider two illustrative scenarios:

Scenario 1: $c < \|g_{ij}(\boldsymbol{\theta})\|, \forall j \in \mathcal{B}_i^t$**.** In this case, all the sample gradients in a batch are clipped. Also, we know that the two sources of randomness (i.e. stochastic and Gaussian noise) are independent, thus their variances can be summed up. Let us assume that $E[\bar{g}_{ij}(\boldsymbol{\theta})] = G_i(\boldsymbol{\theta})$ for all samples j. From Equation (4.1), we can find the mean of each *batch gradient* $\tilde{g}_i^{e,t}(\boldsymbol{\theta})$ (of client i in round e and gradient step t) as follows:

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

Also, from Equation (4.1), we can find the variance of each *batch gradient* $\tilde{g}_i^{e,t}(\boldsymbol{\theta})$ (of client i in round e and gradient step t) as follows:

$$\begin{aligned}
\sigma^2_{i,\tilde{g}}(b_i^e) &:= \texttt{Var}[\tilde{g}_i^{e,t}(\boldsymbol{\theta})] = \texttt{Var}\left[\frac{1}{b_i^e} \sum_{j \in \mathcal{B}_i^{e,t}} \bar{g}_{ij}(\boldsymbol{\theta})\right] + \frac{p\sigma^2_{i,\text{DP}}}{(b_i^e)^2} \\
&= \frac{1}{(b_i^e)^2}\left(\mathbb{E}\left[\left\| \sum_{j \in \mathcal{B}_i^{e,t}} \bar{g}_{ij}(\boldsymbol{\theta})\right\|^2\right] - \left\|\mathbb{E}\left[\sum_{j \in \mathcal{B}_i^{e,t}} \bar{g}_{ij}(\boldsymbol{\theta})\right]\right\|^2\right) + \frac{pc^2 z_i^2(\epsilon_i, \delta_i, b_i^1, b_i^{>1}, N_i, K, E)}{(b_i^e)^2} \\
&= \frac{1}{(b_i^e)^2}\left(\mathbb{E}\left[\left\| \sum_{j \in \mathcal{B}_i^{e,t}} \bar{g}_{ij}(\boldsymbol{\theta})\right\|^2\right] - \left\| \sum_{j \in \mathcal{B}_i^{e,t}} G_i(\boldsymbol{\theta})\right\|^2\right) + \frac{pc^2 z_i^2(\epsilon_i, \delta_i, b_i^1, b_i^{>1}, N_i, K, E)}{(b_i^e)^2} \\
&= \frac{1}{(b_i^e)^2}\Bigg(\underbrace{\mathbb{E}\left[\left\| \sum_{j \in \mathcal{B}_i^{e,t}} \bar{g}_{ij}(\boldsymbol{\theta})\right\|^2\right]}_{\mathcal{A}} - (b_i^e)^2 \left\|G_i(\boldsymbol{\theta})\right\|^2\Bigg) + \frac{pc^2 z_i^2(\epsilon_i, \delta_i, b_i^1, b_i^{>1}, N_i, K, E)}{(b_i^e)^2},
\end{aligned} \tag{C.2}$$

where:

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

The last equation has used Equation (C.1) and that we clip the norm of sample gradients $\bar{g}_{ij}(\boldsymbol{\theta})$ with an "effective" clipping threshold c. By replacing $\mathcal{A}$ into Equation (C.2), we can rewrite it as:

$$\sigma^2_{i,\tilde{g}}(b^e_i) := \texttt{Var}[\tilde{g}^{e,t}_i(\boldsymbol{\theta})] = \frac{1}{(b^e_i)^2}\left(\mathbb{E}\left[\left\|\sum_{j\in\mathcal{B}^{e,t}_i}\bar{g}_{ij}(\boldsymbol{\theta})\right\|^2\right] - (b^e_i)^2\left\|G_i(\boldsymbol{\theta})\right\|^2\right) + \frac{pc^2z^2_i(\epsilon_i,\delta_i,b^1_i,b^{>1}_i,N_i,K,E)}{(b^e_i)^2}$$
$$= \frac{1}{(b^e_i)^2}\left(b^e_i c^2 + \left(2\binom{b^e_i}{2} - (b^e_i)^2\right)\left\|G_i(\boldsymbol{\theta})\right\|^2\right) + \frac{pc^2z^2_i(\epsilon_i,\delta_i,b^1_i,b^{>1}_i,N_i,K,E)}{(b^e_i)^2}$$
$$= \frac{c^2 - \left\|G_i(\boldsymbol{\theta})\right\|^2}{b^e_i} + \frac{pc^2z^2_i(\epsilon_i,\delta_i,b^1_i,b^{>1}_i,N_i,K,E)}{(b^e_i)^2} \approx \frac{pc^2z^2_i(\epsilon_i,\delta_i,b^1_i,b^{>1}_i,N_i,K,E)}{(b^e_i)^2} \tag{C.3}$$

The last approximation is valid because $p \gg 1$ (it is the number of model parameters).

Scenario 2: $c \geq \|g_{ij}(\boldsymbol{\theta})\|, \forall j \in \mathcal{B}^t_i$**.** When the clipping is ineffective for all samples in the batch, we have a noisy version of the batch gradient $g^{e,t}_i(\boldsymbol{\theta}) = \frac{1}{b^e_i}\sum_{j\in\mathcal{B}^{e,t}_i} g_{ij}(\boldsymbol{\theta})$, which is unbiased with variance bounded by $\sigma^2_{i,g}(b^e_i)$. We note that $\sigma^2_{i,g}(b^e_i)$ is a constant that depends on the used batch size b^e_i. The larger the batch size b^e_i used during round e, the smaller the constant. Hence, in this case:

$$\mathbb{E}[\tilde{g}^{e,t}_i(\boldsymbol{\theta})] = \mathbb{E}[g^{e,t}_i(\boldsymbol{\theta})] = \nabla f_i(\boldsymbol{\theta}),$$

and

$$\sigma^2_{i,\tilde{g}}(b^e_i) = \texttt{Var}[\tilde{g}^{e,t}_i(\boldsymbol{\theta})] = \texttt{Var}[g^{e,t}_i(\boldsymbol{\theta})] + \frac{p\sigma^2_{i,\texttt{DP}}}{(b^e_i)^2} \leq \sigma^2_{i,g}(b^e_i) + \frac{p\sigma^2_{i,\texttt{DP}}}{(b^e_i)^2}$$
$$= \sigma^2_{i,g}(b^e_i) + \frac{pc^2z^2_i(\epsilon,\delta,b^1_i,b^{>1}_i,N_i,K,E)}{(b^e_i)^2}$$
$$\approx \frac{pc^2z^2_i(\epsilon,\delta,b^1_i,b^{>1}_i,N_i,K,E)}{(b^e_i)^2}. \tag{C.4}$$

The approximation is valid because $p \gg 1$ (number of model parameters). Also, note that $\sigma^2_{i,g}(b^e_i)$ decreases with b^e_i. Therefore, we got to the same result as in Equation (C.3).

As observed in see Figure 4.4, z_i grows with b^1_i and $b^{>1}_i$ *sub-linearly* (especially with b^1_i). Therefore, the variance of the client i's DP batch gradients $\tilde{g}^{e,t}_i(\boldsymbol{\theta})$ during communication round e, decreases with b^e_i fast. The larger the batch size b^e_i, the less the noise existing in its batch gradients during the same round.

With the findings above, we now investigate the effect of batch size b^e_i on **the noise level in clients' model updates at the end of round** e. During the global communication round e, a participating client i performs $E^e_i = K \cdot \lceil \frac{N_i}{b^e_i} \rceil$ batch gradient updates locally with step size η_l:

$$\boldsymbol{\theta}^{e,k}_i = \boldsymbol{\theta}^{e,k-1}_i - \eta_l \tilde{g}_i(\boldsymbol{\theta}^{e,k-1}_i),\ k = 1,\ldots,E^e_i.$$

Hence,

$$\Delta\tilde{\boldsymbol{\theta}}_i^e = \boldsymbol{\theta}_i^{e,E_i^e} - \boldsymbol{\theta}_i^{e,0}$$

In each update, it adds a Gaussian noise from $\mathcal{N}(0, \frac{c^2 z_i^2(\epsilon,\delta,b^1,b^{>1},N_i,K,E)}{b^{e2}}\mathbb{I}_p)$ to its batch gradients independently (see Equation (4.1)). Hence:

$$\texttt{Var}[\Delta\tilde{\boldsymbol{\theta}}_i^e|\boldsymbol{\theta}_i^{e,0}] = E_i^e \cdot \eta_l^2 \cdot \sigma_{i,\tilde{g}}^2(b_i^e),$$

where $\sigma_{i,\tilde{g}}^2(b_i^e)$ was computed in Equation (C.3) and Equation (C.4), and was a decreasing function of b_i^e. Therefore:

$$\texttt{Var}[\Delta\tilde{\boldsymbol{\theta}}_i^e|\boldsymbol{\theta}_i^{e,0}] \approx K \cdot N_i \cdot \eta_l^2 \cdot \frac{pc^2 z_i^2(\epsilon,\delta,b_i^1,b_i^{>1},N_i,K,E)}{(b_i^e)^3}.$$

□

E.2 Proof of Lemma 5

Lemma 5. *Let $\Delta_{m,m'}(b^1) := \|\mu_m^*(b^1) - \mu_{m'}^*(b^1)\|$ when $\forall i : b_i^1 = b^1$. The overlap between components $\mathcal{N}\big(\mu_m^*(b^1), \Sigma_m^*(b^1)\big)$ and $\mathcal{N}\big(\mu_{m'}^*(b^1), \Sigma_{m'}^*(b^1)\big)$ is $O_{m,m'} = 2Q(\frac{\sqrt{p}\Delta_{m,m'}(b^1)}{2\sigma^1(b^1)})$, where $(\sigma^1(b^1))^2 := \texttt{Var}[\Delta\tilde{\boldsymbol{\theta}}_i^1(b_i^1)|\boldsymbol{\theta}^{init}, b_i^1 = b^1]$ and $Q(\cdot)$ is the Q function. Furthermore, if we increase $b_i^1 = b^1$ to $b_i^1 = kb^1 \leq N$ (for all i), we have $O_{m,m'} \leq 2Q(\frac{\sqrt{kp}\Delta_{m,m'}(b^1)}{2\rho\sigma^1(b^1)})$, where $1 \leq \rho \in \mathcal{O}(1)$ is a small constant.*

Proof. We first find the overlap between two arbitrary Gaussian distributions. Without loss of generality, lets assume we are in 1-dimensional space and that we have two Gaussian distributions both with variance σ^2 and with means $\mu_1 = 0$ and $\mu_2 = \mu$ ($\|\mu_1 - \mu_2\| = \mu$), respectively. Based on symmetry of the distributions, the two components start to overlap at $x = \frac{\mu}{2}$. Hence, we can find the overlap between the two gaussians as follows:

$$O := 2\int_{\frac{\mu}{2}}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} dx = 2\int_{\frac{\mu}{2\sigma}}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx = 2Q(\frac{\mu}{2\sigma}),$$

where $Q(\cdot)$ is the tail distribution function of the standard normal distribution. Now, lets consider the 2-dimensional space, and consider two similar symmetric distributions centered at $\mu_1 = (0,0)$ and $\mu_2 = (\mu,0)$ ($\|\mu_1 - \mu_2\| = \mu$) and with $\Sigma_1 = \Sigma_2 = \begin{bmatrix}\sigma^2 & 0\\ 0 & \sigma^2\end{bmatrix}$. The overlap between the two gaussians can be found as:

$$O = 2\int_{-\infty}^{\infty}\int_{\frac{\mu}{2}}^{\infty}\frac{1}{2\pi\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}}dxdy = 2\int_{\frac{\mu}{2}}^{\infty}\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{x^2}{2\sigma^2}}dx \cdot \int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{y^2}{2\sigma^2}}dy = 2Q(\frac{\mu}{2\sigma}). \tag{C.5}$$

If we compute the overlap for two similar symmetric p-dimensional distributions with $\|\mu_1 - \mu_2\| = \mu$ and variance σ^2 in every direction, we will get to the same result $2Q(\frac{\mu}{2\sigma})$.

In the lemma, when using batch size b^1, we have two Gaussian distributions $\mathcal{N}\big(\mu_m^*(b^1), \Sigma_m^*(b^1)\big)$ and $\mathcal{N}\big(\mu_{m'}^*(b^1), \Sigma_{m'}^*(b^1)\big)$, where

$$\Sigma_m^*(b^1) = \Sigma_{m'}^*(b^1) = \begin{bmatrix} \frac{(\sigma^1(b^1))^2}{p} & & 0 \\ & \ddots & \\ 0 & & \frac{(\sigma^1(b^1))^2}{p} \end{bmatrix}. \tag{C.6}$$

Therefore, from Equation (C.5), we can immediately conclude that the overlap between the two Gaussians, which we denote with $O_{m,m'}(b^1)$, is:

$$O_{m,m'}(b^1) = 2Q(\frac{\sqrt{p}\Delta_{m,m'}(b^1)}{2\sigma^1(b^1)}),$$

which proves the first part of the lemma.

Now, lets see the effect of increasing batch size. First, note that we had:

$$\begin{aligned} &\Delta\tilde{\boldsymbol{\theta}}_i^1 = \boldsymbol{\theta}_i^{1,E_i^1} - \boldsymbol{\theta}_i^{1,0}, \\ &\boldsymbol{\theta}_i^{1,k} = \boldsymbol{\theta}_i^{1,k-1} - \eta_l\tilde{g}_i(\boldsymbol{\theta}_i^{1,k-1}),\ k = 1, \ldots, E_i^1, \end{aligned}$$

where $E_i^1 = K \cdot \lceil\frac{N}{b^1}\rceil$ is the total number of gradients steps taken by client i during communication round $e = 1$. Therefore, considering that DP batch gradients are clipped with a bound c, we have:

$$\|\mathbb{E}[\Delta\tilde{\boldsymbol{\theta}}_i^1(b^1)]\| \leq E_i^1 \cdot \eta_l \cdot c. \tag{C.7}$$

When we increase batch size b_i^1 for all clients from b^1 to kb^1, the upperbound in Equation (C.7) gets k times smaller. In fact by doing so, the number of local gradient updates that client i performs during round $e = 1$, which is equal to E_i^1, decreases k times. As such, we can write:

$$\Delta\tilde{\boldsymbol{\theta}}_i^1(b^1) = k \cdot \Delta\tilde{\boldsymbol{\theta}}_i^1(kb^1) + \upsilon_i,$$

where $\upsilon_i \in \mathbb{R}^p$ is a vector capturing the discrepancies between $\Delta\tilde{\boldsymbol{\theta}}_i^1(b^1)$ and $k \cdot \Delta\tilde{\boldsymbol{\theta}}_i^1(kb^1)$. Therefore, we have:

$$\begin{aligned}\mu_m^*(b^1) &= \mathbb{E}[\Delta\tilde{\boldsymbol{\theta}}_i^1(b^1)|s(i)=m] = \mathbb{E}[k \cdot \Delta\tilde{\boldsymbol{\theta}}_i^1(kb^1) + \upsilon_i|s(i)=m] \\ &= k \cdot \mathbb{E}[\Delta\tilde{\boldsymbol{\theta}}_i^1(kb^1)] + \mathbb{E}[\upsilon_i|s(i)=m] = k \cdot \mu_m^*(kb^1) + \mathbb{E}[\upsilon_i|s(i)=m].\end{aligned}$$

Therefore, we have:

$$\|\mu_m^*(b^1) - \mu_{m'}^*(b^1)\| = \left\| k\mu_m^*(kb^1) - k\mu_{m'}^*(kb^1) + \Big(\mathbb{E}[\upsilon_i|s(i)=m] - \mathbb{E}[\upsilon_i|s(i)=m']\Big)\right\|.$$

Based on our experiments, the last term above, in parenthesis, is small and we can have the following approximation for the equation above:

$$\|\mu_m^*(b^1) - \mu_{m'}^*(b^1)\| \approx \|k\mu_m^*(kb^1) - k\mu_{m'}^*(kb^1)\|,$$

or equivalently:

$$\|\mu_m^*(kb^1) - \mu_{m'}^*(kb^1)\| \approx \frac{\|\mu_m^*(b^1) - \mu_{m'}^*(b^1)\|}{k}. \tag{C.8}$$

Figure E.7 left shows the validity of the approximation above with some experimental results. On the other hand, noting that a client, with dataset size N and batch size b^1, takes $\frac{N}{b^1}$ gradient steps during each epoch of the first round, we have:

$$\begin{aligned}\forall m \in [M] : (\sigma_m^1(b^1))^2 &:= \texttt{Var}[\Delta\tilde{\boldsymbol{\theta}}_i^1|\boldsymbol{\theta}^{init}, b_i^1 = b^1, s(i)=m] \\ &= (\sigma^1(b^1))^2 \approx K \cdot N \cdot \eta_l^2 \cdot \frac{pc^2z^2(\epsilon,\delta,b^1,b^{>1},N,K,E)}{(b^1)^3}.\end{aligned}$$

When we change the batch size used during the first round $e=1$ from b^1 to kb^1 and we fix the batch size of rounds $e>1$, then the noise scale z changes from $z(\epsilon,\delta,b^1,b^{>1},N_i,K,E)$ to $z(\epsilon,\delta,kb^1,b^{>1},N_i,K,E)$. Confirmed by our experimental analysis (see Figure E.7 right), the

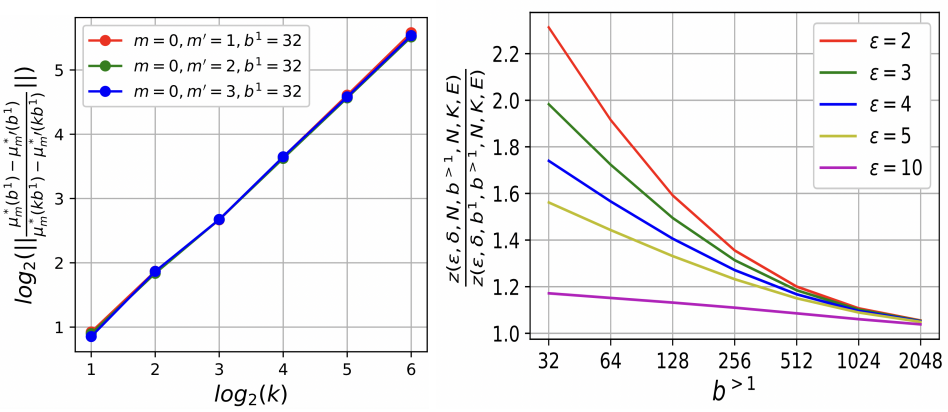


Figure E.7: **Left:** Distance between the centers of different clusters, i.e. the distance between $\mu_m^*(b^1)$ and $\mu_{m'}^*(b^1)$, decreases k times as b^1 increases k times. The three curves in the plot are obtained on CIFAR10 with 4 clusters $m \in \{0,1,2,3\}$ obtained from covariate shift (rotation). The curves are overlapping all with slope 0.95, which is very close to 1. This shows the validity of the approximation in Equation (C.8). **Right:** Effect of changing batch size b^1 to full batch size in the first round on the noise scale z. In the denominator, b^1 is equal to $b^{>1}$. Results are obtained from Rényi-DP accountant (Mironov et al. 2019) with $N = 50000$, $K = 1$ and $E = 200$. For each value of ϵ, we have shown the results for seven values of $b^{>1}$.

amount of change in z due to this is small, as we have changed the batch size only in the first round $e = 1$ from b^1 to kb^1, while the batch sizes in the other $E-1$ rounds are unchanged and $E \gg 1$. Therefore, supported by the results in Figure E.7, we can always establish an upper bound on the amount of change in z as b^1 increases: $z(\epsilon,\delta,kb^1,b^{>1},N,K,E) \leq \rho z(\epsilon,\delta,b^1,b^{>1},N,K,E)$, where ρ is a small constant (e.g., $\rho = 2.5$ in Figure E.7 right). So we have:

$$\begin{aligned}\forall m \in [M] : (\sigma_m^1(kb^1))^2 = (\sigma^1(kb^1))^2 &\approx K \cdot N \cdot \eta_l^2 \cdot \frac{pc^2 z^2(\epsilon,\delta,kb^1,b^{>1},N,K,E)}{(kb^1)^3} \\ &\leq K \cdot N \cdot \eta_l^2 \cdot \frac{pc^2\rho^2 z^2(\epsilon,\delta,b^1,b^{>1},N,K,E)}{(kb^1)^3} \\ &= \frac{\rho^2(\sigma^1(b^1))^2}{k^3}. \end{aligned} \tag{C.9}$$

From Equation (C.8) and Equation (C.9), we have:

$$O_{m,m'}(kb^1) = 2Q\left(\frac{\sqrt{p}\Delta_{m,m'}(kb^1)}{2\sigma^1(kb^1)}\right) \leq 2Q\left(\frac{\sqrt{p}\frac{\Delta_{m,m'}(b^1)}{k}}{2\frac{\rho\sigma^1(b^1)}{k^{\frac{3}{2}}}}\right) = 2Q\left(\frac{\sqrt{kp}\Delta_{m,m'}(b^1)}{2\rho\sigma^1(b^1)}\right),$$

which completes the proof. □

E.3 Proof of Theorem 11

Theorem 11. *(Ma et al. 2000) Given model updates $\{\Delta\tilde{\boldsymbol{\theta}}_i^1(b^1)\}_{i=1}^n$, as samples from a true mixture of Gaussians $\psi^*(b^1) = \{\mathcal{N}\big(\mu_m^*(b^1), \Sigma_m^*(b^1)\big), \alpha_m^*\}_{m=1}^M$, if $O^{max}(\psi^*(b^1))$ is small enough, then:*

$$\lim_{r\to\infty} \frac{\|\psi^{r+1} - \psi^*(b^1)\|}{\|\psi^r - \psi^*(b^1)\|} = o\Big(\big[O^{max}(\psi^*(b^1))\big]^{0.5-\gamma}\Big),$$

as n increases. ψ^r is the parameters returned by EM after r iterations. γ is an arbitrary small positive number, and $o(x)$ means it is a higher order infinitesimal as $x \to 0 : \lim_{x\to 0} \frac{o(x)}{x} = 0$.

Proof. The proof directly follows from the proof of Theorem 1 in Ma et al. 2000 by considering $\{\Delta\tilde{\boldsymbol{\theta}}_i^1(b^1)\}_{i=1}^n$ as the samples of Gaussian mixture $\{\mathcal{N}\big(\mu_m^*(b^1), \Sigma_m^*(b^1)\big), \alpha_m^*\}_{m=1}^M$. □

E.4 Formal privacy guarantees of R-DPCFL

The privacy guarantee of R-DPCFL for each client i in the system comes from the fact that the client runs DPSGD with a fixed DP noise variance $\sigma_{i,\text{DP}}^2 = c^2 \cdot z_i^2(\epsilon, \delta, b_i^1, b_i^{>1}, N_i, K, E)$ in each of its batch gradient computations. We provide a formal privacy guarantee for the algorithm to show the sample-level privacy guarantees provided to each client i against the untrusted server (and any other external third party).

Theorem 13. *The set of model updates $\{\Delta\tilde{\boldsymbol{\theta}}_i^e\}_{e=1}^E$, which are uploaded to the server by each client $i \in \{1, \cdots, n\}$ during the training time, as well as their local model cluster selections satisfy (ϵ, δ)-DP, where the parameters ϵ and δ depend on the amount of DP noise $\sigma_{i,DP}^2$ used by the client.*

Proof. The sensitivity of the batch gradient in eq. (3.1) to every data sample is c. Therefore, based on Proposition 1 mentioned at the beginning of the appendix, each of the batch gradient computations by client i (in the first round $e = 1$ as well as the next rounds $e > 1$) is $(\alpha, \frac{\alpha c^2}{2\sigma_{i,\text{DP}}^2})$-RDP. Therefore, if the client runs ${E_i}^{\text{tot}}$ total number of gradient updates during the training time, which results in the model updates $\{\Delta\tilde{\boldsymbol{\theta}}_i^e\}_{e=1}^E$ uploaded to the server, the set of model updates will be $(\alpha, \frac{{E_i}^{\text{tot}}\alpha c^2}{2\sigma_{i,\text{DP}}^2})$-RDP, according to Theorem 1. Finally, according to Lemma 3, this is equivalent to $(\frac{{E_i}^{\text{tot}}\alpha c^2}{2\sigma_{i,\text{DP}}^2} + \frac{log(1/\delta)}{\alpha-1}, \delta)$-DP (for any $\delta > 0$). The RDP-based guarantee is computed over a set of orders α and the best result among them is chosen as the privacy guarantee. Therefore, the set $\{\Delta\tilde{\boldsymbol{\theta}}_i^e\}_{e=1}^E$ satisfies (ϵ, δ)-DP, with $\epsilon = \frac{{E_i}^{\text{tot}}\alpha c^2}{2\sigma_{i,\text{DP}}^2} + \frac{log(1/\delta)}{\alpha-1}$ derived above, and $\delta > 0$. On the other hand, clients' local cluster selections are also privatized by exponential mechanism and satisfy (ϵ, δ)-DP. Hence, the overall training process for each client satisfies (ϵ, δ)-DP. Tight bounds for the privacy guarantee can be derived by using the numerical procedure, proposed in (Mironov et al. 2019), for accounting sampled Gaussian mechanism. □

F The relation between Lemma 4 and the law of large numbers

We first state the weak law of large numbers and then explain how Lemma 4 is related to it.

Theorem 14 (Weak law of large numbers (Billingsley 1995)). *Suppose that $\{X_i\}_{i=1}^{b}$, is an independent sequence (of size b) of i.i.d random variables with expected value μ and positive variance σ^2. Define $\bar{X}_b = \frac{\sum_{i=1}^{b} X_i}{b}$ as their sample mean. Then, for any positive number $\Delta > 0$:*

$$\lim_{b \to \infty} Pr[|\bar{X}_b - \mu| > \Delta] = 0.$$

In fact, the weak law of large numbers states that the sample mean of some i.i.d random variables converges in probability to their expected value (μ). Furthermore, we can see that $\texttt{Var}[\bar{X}_b] = \frac{\sigma^2}{b}$, which means that *the variance of the sample mean decreases as b increases*.

Now, remember from Equation (4.1) that when computing the DP stochastic batch gradients in round e (with batch size b_i^e), we add DP noise with variance $\sigma^2_{i,\texttt{DP}}/b_i^e$ to each of the b_i^e clipped sample gradients in the batch *and average the resulting b_i^e noisy clipped sample gradients*. The sampled noise terms added to the clipped sample gradients in a batch are i.i.d with mean zero. Therefore, based on the above theorem, the variance of their average over each batch should approach zero as the batch size b_i^e grows. The same discussion applies to all the $K \cdot N_i/b_i^e$ gradient updates performed by client i during a communication round e (whose noises will be summed up), which results in Lemma 4.

G Further related works

Performance parity in FL: Performance parity of the final trained model across clients is an important goal in FL. Addressing this goal, Mohri et al. 2019 proposed Agnostic FL (AFL) by using a min-max optimization approach. TERM (Li et al. 2020a) used tilted losses to up-weight clients with large losses. Finally, Li et al. 2020d and Zhang et al. 2023 proposed q-FFL and PropFair, inspired by α-fairness (Lan et al. 2010) and proportional fairness (Bertsimas et al. 2011), respectively. Generating one common model for all clients, these techniques do not perform well when the data distribution across clients is highly heterogeneous or a structured data heterogeneity exists across clusters of clients. While model personalization techniques (e.g., MR-MTL (Liu et al. 2022)) are proposed for the former case, stronger personalization techniques, e.g., client clustering, are used for the latter.

Differential privacy, group fairness and performance parity: Gradient clipping and random noise addition used in DPSGD disproportionately affect underrepresented groups. Some works tried to address the tension between group fairness and DP in centralized settings (Tran et al. 2020) (by using Lagrangian duality) and FL settings (Pentyala et al. 2022) (by using Secure Multiparty Computation (MPC)). Another work tried to remove the disparate impact of DP on

model performance of minority groups in centralized settings (Esipova et al. 2022), by preventing gradient misalignment across different groups of data. Unlike the previous works on group fairness, our work adopts cross-model fairness, where the utility drop after adding DP must be close for different groups (Dwork et al. 2012), including minority and majority clients. As we consider a structured data heterogeneity across clients, the mentioned approaches are not appropriate, due to generating one single model for all.