Federated Cubic Regularized Newton Learning with Sparsification-amplified Differential Privacy

Wei Huo, Changxin Liu, Kemi Ding, Karl Henrik Johansson, Ling Shi

***Abstract*—This paper investigates the use of the cubic-regularized Newton method within a federated learning framework while addressing two major concerns that commonly arise in federated learning: privacy leakage and communication bottleneck. We introduce a federated learning algorithm called Differentially Private Federated Cubic Regularized Newton (DP-FCRN). By leveraging second-order techniques, our algorithm achieves lower iteration complexity compared to first-order methods. We also incorporate noise perturbation during local computations to ensure privacy. Furthermore, we employ sparsification in uplink transmission, which not only reduces the communication costs but also amplifies the privacy guarantee, resulting in an equivalent level of privacy with reduced noise. We analyze the convergence properties of our algorithm and establish the privacy guarantee. Finally, we conduct experiments on a benchmark dataset to validate the effectiveness of the proposed algorithm.**

***Index Terms*—Federated learning; cubic regularized Newton method; differential privacy; communication sparsification**

I. Introduction

Traditional centralized approaches of training artificial intelligence (AI) models are facing significant challenges due to the emergence of data silos and increasing privacy awareness. In response to these challenges, federated learning (FL) has emerged as a promising approach. FL allows multiple computing devices to collaborate in a distributed paradigm, under the coordination of the cloud, without sharing their local data, to learn a shared large model. FL has found wide applications in domains such as finance [1], healthcare [2], autonomous driving [3], and large language models [4]. The prevailing algorithm used in FL is based on stochastic gradient descent (SGD), usually called Fed-SGD [5], [6]. In this approach, each client trains the model on their local data using SGD and uploads the gradient to the central server. The server then averages all the local gradients and performs a gradient step.

Although first-order methods like SGD are the currently preferred choice for FL, they usually suffer from slow convergence, which can impede the deployment of systems that require fast and efficient processing, such as autonomous vehicles where timely and accurate predictions are vital. Newton's method, a second-order optimization technique, is renowned for its fast convergence for strongly convex and smooth functions in optimization. However, incorporating second-order methods into FL is highly challenging. The key obstacle lies in the non-linear nature of aggregating the solutions from local optimization problems for second-order approximation, unlike the straightforward gradient aggregation. This complexity is evident in recent algorithms like GIANT [7], [8].

While aggregating the local Hessians is possible in theory, uploading the Hessian matrices at each round incurs significant communication costs. Moreover, even without matrix transmission, communication efficiency remains a critical bottleneck in FL, especially when the clients, such as mobile devices, have limited bandwidth. For example, the language model GPT-3 [9] has billions of parameters and therefore is cumbersome to share directly. Traditional first-order optimization methods employ various techniques to improve communication efficiency, such as communication compression [10], variance reduction [11], event-based transmission [12], [13], and partial participation [14]. As for faster second-order approaches, Safaryan et al. [15] have recently introduced a family of federated Newton learning methods that facilitate general contractive compression operators for matrices and partial participation, thereby reducing communication costs. Liu et al. [16] proposed a communication-efficient distributed Newton's method that achieves the super-linear convergence result. Fabbro et al. [17] presented a communication-constrained Newton-type algorithm to accelerate FL.

In addition to slow convergence and communication costs in traditional FL, privacy leakage is another significant concern. That the data is stored locally on the client does not alone offer adequate privacy protection. For instance, recent inference attacks, such as those by Fredrikson et al. [18], Zhu et al. [19], and Liu et al. [20], have demonstrated that sharing local model updates or gradients between clients and the server can result in privacy breaches. Sharing second-order information can also pose privacy concern as it often encompasses the client's data. For example, Yin et al. [21] have shown that the eigenvalues of the Hessian matrix can be utilized to extract critical information from input images. Thus, it is imperative to facilitate FL with a rigorous privacy guarantee. Differential privacy (DP), introduced by Dwork [22], has emerged as the de facto standard among various privacy-preserving frameworks, primarily due to its strong privacy guarantee and effectiveness in data analysis tasks. In differentially private Fed-SGD, the gradient is typically augmented with Gaussian noise to achieve DP [23]–[26]. The number of iterations T influences the noise

W. Huo and L. Shi are with the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong (email: whuoaa@connect.ust.hk, eesling@ust.hk).

C. Liu and K. H. Johansson are with the Division of Decision and Control Systems, School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, and also with Digital Futures, SE-10044 Stockholm, Sweden (email: changxin@kth.se, kallej@kth.se).

K. Ding is with the School of System Design and Intelligent Manufacturing, Southern University of Science and Technology, Shenzhen, 518055, China (email: dingkm@sustech.edu.cn).

level at each iteration due to the composition of DP, with the noise level often being proportional to $\sqrt{T}$. Recently, Ganesh et al. [27] devised a faster differentially private convex optimization approach using second-order methods, attaining (ε, δ)-DP and the utility loss $O(d/\varepsilon^2)$ for d dimensional model. This utility loss bound is optimal, i.e., the best achievable for differentially private optimization [28], [29]. However, this method is restricted to the centralized setting.

Motivated by the above observations, we aim to investigate federated Newton learning while jointly considering DP and communication issues in the algorithm design. Specifically, we propose that each local machine runs a cubic regularized Newton method to update its model, with the addition of noise perturbation during the local computation process. We then allow each client to upload a sparsified update to reduce communication cost. Prior research predominantly considers DP and communication efficiency as separate entities [30], [31]. While some research has explored the joint trade-off among privacy, utility, and communication [32]–[34], they tackled the communication and privacy in a cascaded fashion, i.e., their communication schemes do not directly impact the privacy preservation. Lang et al. [35] recently proposed a method coined joint privacy enhancement and quantization, but they require the server and clients to share a common seed to decode messages. Our study focuses on investigating the interplay between communication and privacy guarantees. We leverage the inherent characteristic of sparsification to amplify the privacy guarantee. Given that local model updates in FL are largely sparse, we propose integrating perturbation with random sparsification to boost privacy preservation. Specifically, sparsification transforms a large vector into a sparse one by keeping only a subset of coordinates while setting other coordinates to zero. This approach reduces the sensitivity of updates to raw data, and results in lower privacy loss during each communication round. We show that the amount of noise needed for DP is directly proportional to the number of transmitted coordinates. This implies that improved communication efficiency can reduce the noise perturbation without compromising privacy. Furthermore, we illustrate that the complexity of our algorithm exhibits an exponential improvement compared to that of first-order methods. As the required noise intensity often depends on the number of iterations, this acceleration further reduces the noise intensity and enhances the balance between privacy and convergence trade-off.

The main contributions of our work are summarized as follows:

1) We develop a differentially private federated cubic regularized Newton learning algorithm (**Algorithm 1**), called DP-FCRN. By using second-order Newton methods, the proposed algorithm achieves rapid convergence. We exploit noise perturbation in local computations to guarantee privacy preservation (**Algorithm 2**). Moreover, we use sparsification to improve communication efficiency. Unlike previous studies that treat DP and communication efficiency as separate goals and neglect the impact of efficient communication on privacy [30]–[34], we use the inherent characteristic of sparsification to enhance privacy.
2) We analyze the impact of sparsification on the utility-privacy trade-off. Specifically, we demonstrate that the required noise magnitude is proportional to the sparsification ratio (**Theorem 1**), which means that the proposed algorithm can exploit sparsified transmission to reduce the magnitude of Gaussian noise. Moreover, we conduct non-asymptotic analysis and obtain the utility loss and the corresponding complexity (**Theorem 2**). The utility loss is optimal and the iteration complexity for achieving the optimal utility loss is an exponential improvement over that of first-order methods.
3) We empirically evaluate our scheme on the benchmark dataset. The experiment results show that our algorithm boosts the model accuracy, and at the same time saves communication costs compared to Fed-SGD under the same DP guarantee.

The remainder of the paper is organized as follows. Preliminaries and the problem formulation are provided in Section II. In Section III, a federated cubic regularized Newton learning algorithm with sparsification-amplified DP is proposed. Then, details on the differential privacy analysis are shown in Section IV and the convergence analysis is presented in Section V. In Section VI, numerical simulations are presented to illustrate the obtained results. Finally, conclusion and future research directions are discussed in Section VII.

Notations: Let $\mathbb{R}^p$ and $\mathbb{R}^{p\times q}$ represent the set of p-dimensional vectors and $p \times q$-dimensional matrices, respectively. $I_p \in \mathbb{R}^{p\times p}$ represents a $p \times p$-dimensional identity matrix. With any positive integer, we denote $[d]$ as the set of integers $\{1, 2, \ldots, d\}$. We use $[\cdot]_j$ to denote the j-th coordinate of a vector and j-th row of a matrix. Let c represent a set of integers, and we denote $[X]_c$ as a vector containing elements $[X]_j$ for $j \in c$ if X is a vector, and as a matrix with row vectors $[X]_j$ for $j \in c$ if X is a matrix. Let $\|\cdot\|$ be the ℓ_2-norm vector norm. For a convex and closed subset $\mathcal{X} \subseteq \mathbb{R}^d$, let $\Pi_{\mathcal{X}} : \mathbb{R}^d \to \mathcal{X}$ be the Euclidean projection operator, given by $\Pi_{\mathcal{X}}(x) = \arg\min_{y\in\mathcal{X}} \|y - x\|$. We use $\mathbb{P}\{\mathcal{A}\}$ to represent the probability of an event $\mathcal{A}$, and $\mathbb{E}[x]$ to be the expected value of a random variable x.

The notation $O(\cdot)$ is used to describe the asymptotic upper bound. Mathematically, $h(n) = O(g(n))$ if there exist positive constants C and n_0 such that $0 \le h(n) \le Cg(n)$ for all $n \ge n_0$. Similarly, the notation $\Omega(\cdot)$ provides the asymptotic lower bound, i.e., $h(n) = \Omega(g(n))$ if there exist positive constants C and n_0 such that $0 \le Cg(n) \le h(n)$ for all $n \ge n_0$. The notation $\tilde{O}(\cdot)$ is a variant of $O(\cdot)$ that ignores logarithmic factors, that is, $h(n) = \tilde{O}(g(n))$ is equivalent to $h(n) = O\left(g(n)log^k n\right)$ for some $k > 0$. The notation $\Theta(\cdot)$ is defined as the tightest bound, i.e., $h(n)$ is said to be $\Theta(g(n))$ if $h(n) = O(g(n))$ and $h(n) = \Omega(g(n))$.

II. Preliminaries and Problem Formulation

This section introduces the fundamental setup of FL along with key concepts on Newton methods with cubic regularization and DP. Subsequently, we outline the considered problem.

A. Basic Setup

We consider a federated setting with n clients and a central server. Each client $i \in [n]$ possesses a private local dataset $\zeta_i = \{\zeta_i^{(1)}, \ldots, \zeta_i^{(m)}\}$ containing a finite set of m data samples. Moreover, each client has a private local empirical risk $f_i(x) = \frac{1}{m}\sum_{j=1}^{m} l(x, \zeta_i^{(j)})$, where $l(x, \zeta_i^{(j)})$ is the loss of model x over the data instance $\zeta_i^{(j)}$ for $j \in [m]$. With the coordination of the central server, all clients aim to train a global model x by solving the following empirical risk minimization problem while maintaining their data locally:

$$\min_{x \in \mathcal{X}} \; f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x), \tag{1}$$

where $\mathcal{X} \subseteq \mathbb{R}^d$ is a convex and closed box constraint. Specifically, the model training process takes place locally on each client, and only the updates are sent to the server for aggregation and global update. The optimal model parameter is defined as $x^* = \arg\min_{x \in \mathcal{X}} f(x)$.

We make the following assumptions on the loss function of the training model:

Assumption 1. *$l(\cdot, \zeta)$ is L_0-Lipschitz, L_1-smooth, and has L_2-Lipschitz Hessian for any ζ.*

Assumption 2. *$l(\cdot, \zeta)$ is μ-strongly convex for any ζ.*

Assumption 3. *$\mathcal{X}$ has a finite diameter D.*

From Assumptions 1 and 2, we infer that also $f_i(\cdot)$ and $f(\cdot)$ are μ-strongly convex, L_0-Lipschitz, L_1-smooth, and have L_2-Lipschitz Hessian.

B. Newton Methods with Cubic Regularization

Newton methods [36] iteratively minimize a quadratic approximation of the function $f(\cdot)$ as

$$\begin{aligned} x_{t+1} = \arg\min_{x \in \mathcal{X}} \Big\{ & f(x_t) + \langle \nabla f(x_t), x - x_t \rangle \\ & + \frac{1}{2}\left\langle \nabla^2 f(x_t)(x - x_t), x - x_t \right\rangle \Big\}. \end{aligned} \tag{2}$$

The Hessian matrix $\nabla^2 f(x_t)$ provides curvature information about $f(\cdot)$ at x_t. Newton methods significantly improve the convergence speed of gradient descent by automatically adjusting the step size along each dimension based on the local curvature at each step.

The cubic regularized Newton method, initially introduced by Nesterov and Polyak [37], incorporates a second-order Taylor expansion with a cubic regularization term. In particular, the update is

$$\begin{aligned} x_{t+1} = \arg\min_{x \in \mathcal{X}} \Big\{ & f(x_t) + \langle \nabla f(x_t), x - x_t \rangle \\ & + \frac{1}{2}\left\langle \nabla^2 f(x_t)(x - x_t), x - x_t \right\rangle \\ & + \frac{L_2}{6}\|x - x_t\|^3 \Big\}, \end{aligned} \tag{3}$$

where L_2 is the Lipschitz Hessian constant in Assumption 1. The cubic upper bound of $f(x_t)$ in (3) serves as a universal upper bound regardless of the specific characteristics of the objective function. However, the function to minimize in each step of (3) does not have a closed-form solution and it is limited to a centralized single node setting, which our algorithm addresses in a federated setting as discussed in Section III.

C. Threat Model and DP

Local datasets typically contain sensitive user information. If problem (1) is addressed in an insecure environment, the leakage of information could pose a threat to personal and property privacy. This paper considers the following adversary model [38]:

Definition 1. *(Adversary Model) Adversaries can be*

i) *the honest-but-curious central server that follows the designed training protocol but may be curious about clients' private data and capable of inferring it from shared messages.*
ii) *certain clients colluding with the central server or amongst themselves to deduce private information about a specific client victim.*
iii) *an outside eavesdropper who can intercept all shared messages in the execution of the training protocol but will not actively inject false messages into or interrupt message transmissions. geyer2017differentially*

Our considered adversary model is much stronger than some works that require a trusted third party [39], [40].

DP is a privacy concept widely adopted for quantifying privacy risk. It is a characteristic of a randomized algorithm $\mathcal{A}$ where the presence or absence of an individual in a dataset cannot be determined from the output of $\mathcal{A}$ [22]. Here, we present the formal definition of DP within the context of FL.

Definition 2. *((ε, δ)-DP) The algorithm $\mathcal{A}$ is called (ε, δ)-DP, if for any neighboring dataset pair $\zeta = \cup_{i\in[n]}\zeta_i$ and $\zeta' = \cup_{i\in[n]}\zeta_i'$ that differ in one data instance and every measurable $\mathcal{O} \subseteq \mathrm{Range}(\mathcal{A})$ [1], the output distribution satisfies*

$$\mathbb{P}\{\mathcal{A}(\zeta) \in \mathcal{O}\} \le e^{\varepsilon}\mathbb{P}\{\mathcal{A}(\zeta') \in \mathcal{O}\} + \delta, \tag{4}$$

where the probability $\mathbb{P}\{\cdot\}$ is taken over the randomness of $\mathcal{A}$.

Definition 2 states that the output distributions of neighboring datasets exhibit small variation. The factor ε in (4) represents the upper bound of privacy loss by algorithm $\mathcal{A}$, and δ denotes the probability of breaking this bound. Therefore, a smaller ε corresponds to a stronger privacy guarantee. The Gaussian mechanism is one of the commonly employed techniques to achieve DP.

Lemma 1. *(Gaussian Mechanism [41]) A Gaussian mechanism $\mathcal{G}$ for a vector-valued computation $r : \zeta \to \mathbb{R}^d$ is obtained by computing the function r on the input data $\zeta_i \in \zeta$ and then adding random Gaussian noise perturbation $\nu \sim \mathcal{N}(0, \sigma^2 I_d)$ to the output, i.e,*

$$\mathcal{G} = r(\zeta) + \nu.$$

[1]Range($\mathcal{A}$) denotes the set of all possible observation sequences under the algorithm $\mathcal{A}$.

The Gaussian mechanism $\mathcal{G}$ is $\left(\frac{\sqrt{2\log(1.25/\delta)}\Delta}{\sigma}, \delta\right)$-DP for any neighboring dataset ζ and ζ', where Δ denotes the sensitivity of r, i.e., $\Delta = \sup_{\zeta,\zeta'} \|r(\zeta) - r(\zeta')\|$.

Lemma 1 indicates that achieving (ε, δ)-DP requires adjusting the noise intensity based on the privacy guarantee ε and δ, as well as the sensitivity Δ.

D. Problem Statement

This paper aims to answer the following questions:

(a) How can we develop a cubic regularized Newton algorithm for solving (1) in a federated setting?

(b) Can we explore the sparsification scheme to reduce communication costs while amplifying the privacy guarantee, i.e., achieving a smaller ε given σ or requiring a smaller σ given ε?

(c) What level of noise intensity, i.e., σ, is necessary to attain (ε, δ)-DP in the proposed algorithm?

(d) Is it possible to attain the optimal utility loss under DP, i.e., $f(x_T) - f(x^*) = O(d/\varepsilon^2)$ with the output x_T? If achievable, what is the iteration complexity for achieving this optimal utility loss?

III. Main Algorithm

In this section, we present Algorithms 1 and 2 to answer problem (a) and (b) in Section II-D.

In general, there are two approaches for integrating sparsification and privacy in FL: (1) perturb first, then sparsify, and (2) sparsify first, then perturb. The first approach is straightforward and versatile as sparsification maintains DP and seamlessly integrates with any existing privacy mechanisms. However, in the second approach, perturbation may compromise the communication savings achieved through sparsification. Furthermore, empirical observations suggest that the first approach outperforms the second one in some scenarios [42]. Therefore, we adopt the first approach in this study.

As shown in Algorithm 1, during iteration t, the server broadcasts the parameter x_t to the clients. Each client i randomly samples a data instance $\zeta_{i,t} \in \zeta_i$, estimates the local gradient $\hat{g}_{i,t} = \nabla l(x_t, \zeta_{i,t})$ and the local Hessian $\hat{H}_{i,t} = \nabla^2 l(x_t, \zeta_{i,t})$ to minimize a local cubic-regularized auxiliary loss function based on its local data, and then does the following update

$$x_{i,t+1} = \arg\min_{x\in\mathcal{X}} \Big\{ f_i(x_t) + \langle \hat{g}_{i,t}, x - x_t\rangle + \frac{1}{2}\left\langle \hat{H}_{i,t}(x - x_t), x - x_t\right\rangle + \frac{L_2}{6}\|x - x_t\|^3 \Big\}. \tag{5}$$

As there is no closed form for optimal solution to (5), the client instead employs the gradient descent method to compute $x_{i,t+1}$. To privately minimize the local cubic upper bound, Gaussian noise is added to perturb the gradient. This local solver utilizing the Gaussian mechanism is denoted GMSolver and is detailed in Algorithm 2.

Following the update of the local model parameter, each client uploads its model update $x_{i,t+1} - x_t$ to the server. To address the communication challenges in uplink transmissions, the random-k sparsifier is employed to reduce the message size by a factor of k/d [43]:

Algorithm 1 DP-FCRN

```
Input: Clients' data $\zeta_1, \dots, \zeta_n$, sparsification parameter $k$, DP parameters $(\varepsilon, \delta)$, and scaling factor $\alpha$.
Initialize: Model parameter $x_0$.
1:  for $t = 0, 1, \dots, T-1$ do
2:      ▶ Server broadcasts
3:      Broadcast $x_t$ to all clients
4:      ▶ Clients update and upload
5:      for each client $i \in [n]$ in parallel do
6:          Sample $\zeta_{i,t}$ uniformly from $\{\zeta_i^{(1)}, \dots, \zeta_i^{(m)}\}$ and compute the local estimate gradient $\hat{g}_{i,t} = \nabla l(x_t, \zeta_{i,t})$ and the local estimate Hessian $\hat{H}_{i,t} = \nabla^2 l(x_t, \zeta_{i,t})$
7:          $x_{i,t+1} = \mathrm{GMSolver}(x_t, \hat{g}_{i,t}, \hat{H}_{i,t}, \tau, \sigma)$
8:          $y_{i,t} \leftarrow \frac{x_{i,t+1} - x_t}{\alpha}$ and upload $\mathcal{S}(y_{i,t})$ to the server
9:      end for
10:     ▶ Server updates
11:     $x_{t+1} = x_t + \frac{\alpha}{n}\sum_{i\in\mathcal{I}_t} \mathcal{S}(y_{i,t})$
12: end for
```

Algorithm 2 GMSolver

```
Input: Initialization $\theta_0$, gradient $g$, Hessian $H$, the number of iterations $\tau$, and the noise parameter $\sigma$.
1: for $s = 0, 1, \dots, \tau - 1$ do
2:     $\eta_s = \frac{2}{\mu(s+2)}$
3:     $\mathrm{grad}_s = g + H(\theta_s - \theta_0) + \frac{L_2}{2}\|\theta_s - \theta_0\|(\theta_s - \theta_0)$
4:     $\theta_{s+1} = \Pi_{\mathcal{X}}[\theta_s - \eta_s(\mathrm{grad}_s + b_s)]$, where $b_s \sim N(0, \sigma^2 I_d)$
5: end for
6: Return $\sum_{s=0}^{\tau-1} \frac{2(s+1)}{\tau(\tau+1)}\theta_s$
```

Definition 3. *(Random-k Sparsification): For $x \in \mathbb{R}^d$ and a parameter $k \in [d]$, the random-k sparsification operator is defined as*

$$\mathcal{S}(x) := \frac{d}{k}(\xi_k \odot x),$$

where $\odot$ denotes the Hadamard (element-wise) product and $\xi_k \in \{0, 1\}^d$ is a uniformly random binary vector with k nonzero entries, i.e., $\|\xi_k\|_0 = k$.

Integrating private GMSolver and random-k sparsification, the proposed algorithm simultaneously addresses privacy preservation and communication efficiency as depicted in Algorithm 1. A scaling factor $\alpha > 0$ is introduced for convergence analysis.

Remark 1. *As pointed out by Lacoste-Julien et al. [44], the output of Algorithm 2 can be computed online. Specifically, setting $z_0 = \theta_0$, and recursively defining $z_s = \rho_s\theta_s + (1 -$*

$\rho_s)z_{s-1}$ for $s \geq 1$, with $\rho_s = \frac{2}{s+1}$. It is a straightforward calculation to check that $z_\tau = \sum_{s=0}^{\tau-1} \frac{2s}{\tau(\tau+1)}\theta_s$.

IV. Privacy Analysis

In this section, we prove the privacy guarantee provided by Algorithm 1.

To establish the privacy-preserving property of the proposed algorithm, we make the following assumption.

Assumption 4. *For any data sample $\zeta_i^{(j)} \in \zeta_i$ and $h \in [d]$, we have*

$$\left|\left[\nabla l(x,\zeta_i^{(j)})\right]_h\right| \leq \frac{G_1}{\sqrt{d}}, \quad \left\|\left[\nabla^2 l(x,\zeta_i^{(j)})\right]_h\right\| \leq \frac{G_2}{\sqrt{d}}$$

for any $x, v \in \mathcal{X}$ and $i \in [n]$.

Assumption 4 characterizes the sensitivity of each coordinate of gradient $\nabla l(x,\zeta_i^{(j)})$ and each row of Hessian $\nabla^2 l(x,\zeta_i^{(j)})$, and implies $\left\|\nabla l(x,\zeta_i^{(j)})\right\| \leq G_1$ and $\left\|\nabla^2 l(x,\zeta_i^{(j)})\right\|_2 \leq \left\|\nabla^2 l(x,\zeta_i^{(j)})\right\|_F \leq G_2$, which can be enforced by the gradient and Hessian clip techniques [27].

To analyze the interplay between the sparsification and privacy, let c_i^t denote the randomly selected coordinate set for participating client i at round t, i.e., $\mathcal{S}(\cdot) = \frac{d}{k}[\cdot]_{c_i^t}$. An important observation is that only the values in c_i^t are transmitted to the central server, i.e.,

$$\mathcal{S}(y_{i,t}) = \frac{d}{k}\left[\frac{x_{i,t+1}-x_{i,t}}{\alpha}\right]_{c_i^t} = \frac{d}{\alpha k}\left([x_{i,t+1}]_{c_i^t} - [x_{i,t}]_{c_i^t}\right).$$

The gradient update information is contained in $[x_{i,t+1}]_{c_i^t}$ and

$$[x_{i,t+1}]_{c_i^t} = \left[\sum_{s=0}^{\tau-1}\frac{2(s+1)}{\tau(\tau+1)}\theta_i^{t,s}\right]_{c_i^t} = \sum_{s=0}^{\tau-1}\frac{2(s+1)}{\tau(\tau+1)}[\theta_i^{t,s}]_{c_i^t},$$

where $\theta_i^{t,s}$ denotes the optimization variable used by client i at iteration s in Algorithm 2 and the communication round t in Algorithm 1. Based on step 4 in the GMSolver, we have

$$[\theta_i^{t,s+1}]_{c_i^t} = \left[\Pi_{\mathcal{X}}\left[\theta_i^{t,s} - \eta_s(\text{grad}_i^{t,s} + b_i^{t,s})\right]\right]_{c_i^t},$$

where $\text{grad}_i^{t,s}$ and $b_i^{t,s}$ are the gradient and noise used by client i at iteration s in GMSolver and the communication round t in Algorithm 1, respectively. Since projection into a box constraint does not influence the set of selected coordinators c_i^t, what matters in local computation is

$$\left[\theta_i^{t,s} - \eta_s(\text{grad}_i^{t,s} + b_i^{t,s})\right]_{c_i^t} = [\theta_i^{t,s}]_{c_i^t} - \eta_s[\text{grad}_i^{t,s} + b_i^{t,s}]_{c_i^t}.$$

According to the above analysis, we conclude that the crucial aspect of privacy protection lies in the sparsified noisy gradient update, which can be expressed as

$$[\text{grad}_i^{t,s} + b_i^{t,s}]_{c_i^t} = [\text{grad}_i^{t,s}]_{c_i^t} + [b_i^{t,s}]_{c_i^t}.$$

We observe that due to the sparsification, Gaussian noises are only added to the values of active coordinates in c_i^t. If noise is added only at the selected coordinates, the level of privacy remains the same. In other words, the same level of privacy assurance can be maintained even when incorporating a diminished amount of additional noise, thereby enhancing the optimization accuracy. Therefore, we need to analyze the privacy loss of $[\text{grad}_i^{t,s}]_{c_i^t}$ after adding noise $[b_i^{t,s}]_{c_i^t}$.

For client i, considering any two neighboring dataset ζ_i and ζ_i' of the same size m but differ only in one data sample (e.g., $\zeta_i^{j_0}$ and $\zeta_i^{j_0\prime}$), let Δ be the ℓ_2-sensitivity of $[\text{grad}_i^{t,s}]_{c_i^t}$. Then we have

$$\begin{aligned}
&\Delta^2 \\
=&\max_{\zeta,\zeta'}\Big\| [\hat{g}_{i,t}]_{c_i^t} - [\hat{g}_{i,t}']_{c_i^t} + \left[\hat{H}_{i,t}(\theta_i^{t,s}-x_t)\right]_{c_i^t} \\
&\qquad - \left[\hat{H}_{i,t}'(\theta_i^{t,s}-x_t)\right]_{c_i^t}\Big\|^2 \\
=&\max_{\zeta,\zeta'}\Big\| \left[\nabla l(x_t,\zeta_i^{j_0}) - \nabla l(x_t,\zeta_i^{j_0\prime})\right]_{c_i^t} \\
&\qquad + \left[(\nabla^2 l(x_t,\zeta_i^{j_0}) - \nabla^2 l(x_t,\zeta_i^{j_0\prime}))(\theta_i^{t,s}-x_t)\right]_{c_i^t}\Big\|^2 \\
\leq&\frac{4k(G_1+G_2D)^2}{d},
\end{aligned} \tag{6}$$

where the last inequality holds from

$$\begin{aligned}
&\Big\| \left[\nabla l(x_t,\zeta_i^{j_0}) - \nabla l(x_t,\zeta_i^{j_0\prime})\right]_{c_i^t} \\
&\qquad + \left[(\nabla^2 l(x_t,\zeta_i^{j_0}) - \nabla^2 l(x_t,\zeta_i^{j_0\prime}))(\theta_i^{t,s}-x_t)\right]_{c_i^t}\Big\| \\
\leq&\left\|\left[\nabla l(x_t,\zeta_i^{j_0}) - \nabla l(x_t,\zeta_i^{j_0\prime})\right]_{c_i^t}\right\| \\
&+ \left\|\left[\nabla^2 l(x_t,\zeta_i^{j_0}) - \nabla^2 l(x_t,\zeta_i^{j_0\prime})\right]_{c_i^t}[\theta_i^{t,s}-x_t]_{c_i^t}\right\| \\
\leq&\frac{2\sqrt{k}G_1}{\sqrt{d}} + \frac{2\sqrt{k}G_2D}{\sqrt{d}}.
\end{aligned}$$

Lemma 1 indicates that the required noise intensity for achieving (ε,δ)-DP relies on the sensitivity value. From (6), sparsification reduces the conventional sensivity $2(G_1+G_2D)$ by a factor of $\sqrt{k/d}$. Consequently, sparsification decreases sensitivity, leading to a reduction in noise intensity. Theorem 1 states a sufficient condition for achieving (ε,δ)-DP based on the reduced sensitivity resulting from sparsification.

Theorem 1. *Suppose that Assumptions 1 and 3 are satisfied, and the random-k sparsifier with $k \leq d$ is used in Algorithm 1. Given parameters m, τ, $\varepsilon \in (0,1]$, and $\delta_0 \in (0,1]$, if*

$$\sigma^2 \geq \frac{80\tau T k\log(1.25/\delta_0)(G_1+G_2D)^2}{\varepsilon^2 m^2 d} \tag{7}$$

and $T \geq \frac{\varepsilon^2}{4\tau}$, then Algorithm 1 is (ε,δ)-DP for some constant $\delta \in (0,1]$.

Proof. The proof is provided in Appendix B. □

Remark 2. *The required noise intensity is proportional to the sparsification ratio, k/d. Therefore, to achieve the same level of DP, the required noise under our algorithm can be reduced by decreasing k. In other words, the fewer transmitted bits, the less noise required for (ε,δ)-DP.*

V. Convergence Analysis

This section presents the convergence analysis of Algorithm 1. We begin by introducing some properties of random-k sparsification.

Lemma 2. *The random-k sparsification operator $\mathcal{S}(x)$ exhibits the following properties:*

$$\mathbb{E}[\mathcal{S}(x)] = x,\ \mathbb{E}\left[\|\mathcal{S}(x) - x\|^2\right] \leq \left(\frac{d}{k} - 1\right)\|x\|^2.$$

In each step of the algorithm, a global cubic upper bound function $\phi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ for $f(w)$ is constructed as

$$\begin{aligned}&\phi(v; w)\\ \triangleq& f(w) + \langle \nabla f(w), v - w\rangle + \frac{1}{2}\left\langle \nabla^2 f(w)(v-w), v-w\right\rangle\\ &+ \frac{M}{6}\|v-w\|^3, \quad \forall v \in \mathcal{X},\end{aligned}$$

and local cubic upper bound functions $\phi_i : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ for $f_i(w)$, $i \in \{1, 2, \ldots, n\}$, as

$$\begin{aligned}&\phi_i(v; w)\\ \triangleq& f_i(w) + \langle \nabla f_i(w), v - w\rangle + \frac{1}{2}\left\langle \nabla^2 f_i(w)(v-w), v-w\right\rangle\\ &+ \frac{M}{6}\|v-w\|^3, \quad \forall v \in \mathcal{X}. \qquad (8)\end{aligned}$$

Algorithm 2 uses the typical SGD to solve (5) and the local cubic upper bound $\phi_i(x; x_t)$ is a strongly convex function. Therefore, we can obtain the suboptimality gap based on SGD analysis.

Lemma 3. *Suppose that Assumptions 1 and 3 hold. For every $\beta \in (0, 1)$, given parameters $\varepsilon \in (0, 1]$, $\delta_0 \in (0, 1]$, and $w \in \mathcal{X}$ the output of Algorithm 2, if we set the number of local iterations as*

$$\tau = \frac{(L_0 + L_1 D + MD^2/2)^2 \varepsilon^2 m^2}{Tk\log(1/\delta_0)}, \qquad (9)$$

and the noise as (7), then $\hat{v}$ satisfies

$$\begin{aligned}&\mathbb{E}[\phi_{i,t}(\hat{v}; w)] - \min_{v\in\mathcal{X}} \phi_{i,t}(v; w)\\ =& O\left(\frac{k\log(1/\delta_0)(G_1 + G_2 D)^2 T}{\varepsilon^2 m^2 \mu}\right). \qquad (10)\end{aligned}$$

Proof. The proof is provided in Appendix C. □

Lemma 3 quantifies the suboptimal gap when solving (5) with Algorithm 2 for each client in every communication round. Based on this result, we are in a position to provide the convergence of Algorithm 1.

Theorem 2. *Suppose that Assumptions 1 and 3 hold and the random-k sparsifier with $k \leq d$ is used in Algorithm 1. Then for every $\beta \in (0, 1)$, given parameters m and $\varepsilon \in (0, 1]$, $\delta_0 \in (0, 1]$, by setting the number of local iterations as (9), the scale factor as*

$$\alpha = \frac{k\log(1/\delta_0)(G_1 + G_2 D)^2 T}{\varepsilon^2 m^2 \mu (2L_0 + 2L_1 D + MD^2) D},$$

and the number of iterations in Algorithm 1 to

$$T = \Theta\left(\frac{\sqrt{L_2}(f(x_0) - f(x^*))^{\frac{1}{4}}}{\mu^{\frac{3}{4}}} + \log\log\left(\frac{\varepsilon m}{\sqrt{k\log(1/\delta_0)}}\right)\right),$$

the output of Algorithm 1, i.e., x_T, satisfies (ε, δ)-DP and

$$\begin{aligned}&\mathbb{E}[f(x_T)] - f(x^*)\\ \leq& \tilde{O}\left(\frac{k\log(1/\delta_0)(G_1 + G_2 D)^2}{\varepsilon^2 m^2 \mu} \cdot \left(\frac{L_2^2 L_0 D}{\mu^3}\right)^{\frac{1}{4}}\right).\end{aligned}$$

Proof. The proof is provided in Appendix D. □

Remark 3. *The lower bound on the optimization error of any DP algorithm for the class of strongly convex functions implies that the achievable optimization in Theorem 2 is optimal in terms of the dependence on the number of transmitted coordinates k and the privacy loss ε [28], [29]. Also, the oracle complexity of our algorithm is an exponential improvement over that of first-order methods.*

Remark 4. *The proof of Theorem 2 suggests that DP-FCRN has two phases. First, when x_t is far from x^*, the convergence rate is $1/T^4$. Second, when x_t is close to x^*, the algorithm exhibits the convergence rate of $\exp(\exp(-T))$. Notice that DP-FCRN is agnostic to this transition in the sense that we do not have an explicit switching step. It is interesting to note that the transition happens when $\|x_t - x^*\| \leq 3\mu/4L_2$.*

VI. Numerical Evalucation

In this section, we evaluate the performance of DP-FCRN. We aim to evaluate the performance of DP-FCRN with different levels of compression and compare them with the first-order Fed-SGD.

A. Experimental Setup

We use the benchmark datasets *epsilon* [45] in the experiments. This dataset consists of 400,000 samples and each sample has 2,000 features. We randomly assign the data samples evenly among the $n = 40$ clients. The clients aim to solve the following logistic regression problem:

$$\min_{x\in\mathcal{X}} f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x),$$

where

$$f_i(x) = \frac{1}{m}\sum_{j=1}^{m} \log(1 + \exp(-b_j a_j^\top x)) + \frac{1}{2m}\|x\|^2,$$

$\mathcal{X} \subseteq [-0.5, 0.5]^d$, $a_j \in \mathbb{R}^d$ and $b_j \in \{-1, 1\}$ are the data samples and m denotes the number of samples in the local dataset.

As DP parameters, we consider $\varepsilon \in \{0.4, 0.6, 0.8, 1\}$ and $\delta_0 = 0.01$. The random noise is generated according to (7) and the number of local iterations τ is determined by (9). In iteration t, client i processes one data point from ζ_i and

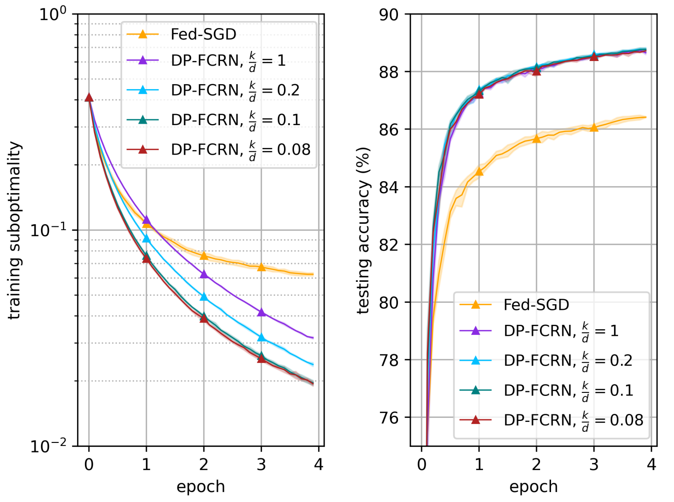

Fig. 1. Performance comparison between Fed-SGD with DP and DP-FCRN with $\varepsilon = 0.8$.

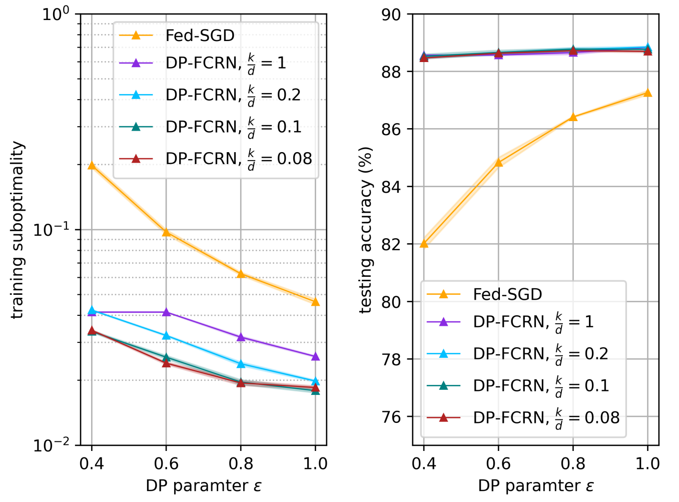

Fig. 2. Performance comparison between Fed-SGD and DP-FCRN under different DP parameters.

the server updates x_t accordingly. Upon finishing processing the entire dataset, one epoch is completed. We conduct the algorithm for four epochs, and repeat each experiment five times. We depict the mean curve along with the region representing one standard deviation. The convergence performance of the algorithm is assessed through training suboptimality and testing accuracy over iterations. Training suboptimality is calculated by $f(x_t) - f(x^*)$, where the ground truth $f(x^*)$ is obtained using the LogisticSGD optimizer from scikit-learn [46]. Testing accuracy is determined by applying the logistic function to the entire dataset. It is calculated as the percentage of correct predictions out of the total number of predictions.

B. Performance and Comparison with Fed-SGD

We set $\varepsilon = 0.8$ and compare the convergence performance between first-order Fed-SGD with DP and Algorithm 1 with different choices of sparsification ratio $k/d \in \{0.08, 0.1, 0.2, 1\}$. Fig. 1 shows that Algorithm 1 outperforms Fed-SGD with DP [47] in terms of convergence speed and model accuracy. Furthermore, the use of a larger sparsification ratio k/d in Algorithm 1 leads to higher training suboptimality, which verifies Theorem 2. We observe that selecting a higher number of retained coordinates in sparsification leads to more complete information transmission as well as increased noise. The findings from Fig. 1 indicate that, in this specific example, the impact of increased noise on convergence performance may outweigh the benefits of enhanced information completeness. On the other hand, there is no significant difference in testing accuracy with different sparse ratios, which indicates that the performance under the proposed DP-FCRN does not deteriorate much while reducing the communication burden.

C. Trade-off between Privacy and Utility

Fig. 2 illustrates the trade-off between privacy and utility. It shows that increasing the value of ε, i.e., a less stringent privacy requirement, results in decreased utility loss across all the methods. Additionally, the performance between Algorithm 1 and Fed-SGD is more significant for the case with smaller ε, i.e., a tighter DP requirement.

VII. Conclusion and Future Work

This paper explores communication efficiency and differential privacy within federated second-order methods. We demonstrate that the inherent sparsification characteristic can bolster the privacy protection. Moreover, employing second-order methods in a privacy setting can achieve the worst-case convergence guarantees and a faster convergence rate. Experiment results illustrate that our algorithm substantially outperforms first-order Fed-SGD in terms of utility loss.

There are several promising directions for future research. Firstly, investigating methods to reduce the computational complexity of federated second-order learning approaches is valuable. Additionally, integrating more general compression schemes and studying the privacy preservation and performance of non-convex cost functions are intriguing topics.

References

[1] David Byrd and Antigoni Polychroniadou. Differentially private secure multi-party computation for federated learning in financial applications. In *Proceedings of the First ACM International Conference on AI in Finance*, pages 1–9, 2020.
[2] Saurabh Singh, Shailendra Rathore, Osama Alfarraj, Amr Tolba, and Byungun Yoon. A framework for privacy-preservation of IoT healthcare data using federated learning and blockchain technology. *Future Generation Computer Systems*, 129:380–388, 2022.
[3] Anh Nguyen, Tuong Do, Minh Tran, Binh X Nguyen, Chien Duong, Tu Phan, Erman Tjiputra, and Quang D Tran. Deep federated learning for autonomous driving. In *IEEE Intelligent Vehicles Symposium*, pages 1824–1830, 2022.
[4] Gyunyeop Kim, Joon Yoo, and Sangwoo Kang. Efficient federated learning with pre-trained large language model using several adapter mechanisms. *Mathematics*, 11(21):4479, 2023.
[5] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1310–1321, 2015.
[6] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

[7] Shusen Wang, Fred Roosta, Peng Xu, and Michael W Mahoney. Giant: Globally improved approximate newton method for distributed optimization. *Advances in Neural Information Processing Systems*, 31, 2018.

[8] Alessio Maritan, Ganesh Sharma, Luca Schenato, and Subhrakanti Dey. Network-GIANT: Fully distributed newton-type optimization via harmonic hessian consensus. *arXiv preprint arXiv:2305.07898*, 2023.

[9] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

[10] Peter Richtárik, Igor Sokolov, and Ilyas Fatkhullin. Ef21: A new, simpler, theoretically better, and practically faster error feedback. *Advances in Neural Information Processing Systems*, 34:4384–4396, 2021.

[11] Bin Wang, Jun Fang, Hongbin Li, and Bing Zeng. Communication-efficient federated learning: A variance-reduced stochastic approach with adaptive sparsification. *IEEE Transactions on Signal Processing*, 71:3562–3576, 2023.

[12] Xingkang He, Xinlei Yi, Yanlong Zhao, Karl Henrik Johansson, and Vijay Gupta. Asymptotic analysis of federated learning under event-triggered communication. *IEEE Transactions on Signal Processing*, 71:2654–2667, 2023.

[13] Bin Wang, Jun Fang, Hongbin Li, and Yonina C. Eldar. Communication efficient confederated learning: An event-triggered SAGA approach. *IEEE Transactions on Signal Processing*, pages 1–17, 2024.

[14] Shiqiang Wang and Mingyue Ji. A unified analysis of federated learning with arbitrary client participation. *Advances in Neural Information Processing Systems*, 35:19124–19137, 2022.

[15] Mher Safaryan, Rustem Islamov, Xun Qian, and Peter Richtárik. FedNL: Making Newton-type methods applicable to federated learning. *arXiv preprint arXiv:2106.02969*, 2021.

[16] Huikang Liu, Jiaojiao Zhang, Anthony Man-Cho So, and Qing Ling. A communication-efficient decentralized Newton's method with provably faster convergence. *IEEE Transactions on Signal and Information Processing over Networks*, 9:427–441, 2023.

[17] Nicolò Dal Fabbro, Subhrakanti Dey, Michele Rossi, and Luca Schenato. SHED: A Newton-type algorithm for federated learning based on incremental Hessian eigenvector sharing. *Automatica*, 160:111460, 2024.

[18] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333, 2015.

[19] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. *Advances in Neural Information Processing Systems*, 32, 2019.

[20] Lan Liu, Yi Wang, Gaoyang Liu, Kai Peng, and Chen Wang. Membership inference attacks against machine learning models via prediction sensitivity. *IEEE Transactions on Dependable and Secure Computing*, 2022.

[21] Xiaoxia Yin, Brian WH Ng, Jing He, Yanchun Zhang, and Derek Abbott. Accurate image analysis of the retina using hessian matrix and binarisation of thresholded entropy with application of texture mapping. *PloS one*, 9(4):e95943, 2014.

[22] Cynthia Dwork. Differential privacy. In *International Colloquium on Automata, Languages, and Programming*, pages 1–12. Springer, 2006.

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

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

[25] Antonious Girgis, Deepesh Data, Suhas Diggavi, Peter Kairouz, and Ananda Theertha Suresh. Shuffled model of differential privacy in federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2521–2529. PMLR, 2021.

[26] Jiaojiao Zhang, Dominik Fay, and Mikael Johansson. Dynamic privacy allocation for locally differentially private federated learning with composite objectives. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 9461–9465, 2024.

[27] Arun Ganesh, Mahdi Haghifam, Thomas Steinke, and Abhradeep Guha Thakurta. Faster differentially private convex optimization via second-order methods. *Advances in Neural Information Processing Systems*, 36, 2024.

[28] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 464–473, 2014.

[29] Peter Kairouz, Ziyu Liu, and Thomas Steinke. The distributed discrete gaussian mechanism for federated learning with secure aggregation. In *International Conference on Machine Learning*, pages 5201–5212. PMLR, 2021.

[30] Zhize Li, Haoyu Zhao, Boyue Li, and Yuejie Chi. Soteriafl: A unified framework for private federated learning with communication compression. *Advances in Neural Information Processing Systems*, 35:4285–4300, 2022.

[31] Xin Zhang, Minghong Fang, Jia Liu, and Zhengyuan Zhu. Private and communication-efficient edge learning: a sparse differential gaussian-masking distributed SGD approach. In *Proceedings of the Twenty-First International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, pages 261–270, 2020.

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

[33] Nima Mohammadi, Jianan Bai, Qiang Fan, Yifei Song, Yang Yi, and Lingjia Liu. Differential privacy meets federated learning under communication constraints. *IEEE Internet of Things Journal*, 9(22):22204–22219, 2021.

[34] Wei-Ning Chen, Christopher A Choquette-Choo, and Peter Kairouz. Communication efficient federated learning with secure aggregation and differential privacy. In *NeurIPS Workshop Privacy in Machine Learning*, 2021.

[35] Natalie Lang, Elad Sofer, Tomer Shaked, and Nir Shlezinger. Joint privacy enhancement and quantization in federated learning. *IEEE Transactions on Signal Processing*, 71:295–310, 2023.

[36] Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[37] Yurii Nesterov and Boris T Polyak. Cubic regularization of Newton method and its global performance. *Mathematical Programming*, 108(1):177–205, 2006.

[38] Lingjuan Lyu, Han Yu, and Qiang Yang. Threats to federated learning: A survey, 2020.

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

[40] Meng Hao, Hongwei Li, Xizhao Luo, Guowen Xu, Haomiao Yang, and Sen Liu. Efficient and privacy-enhanced federated learning for industrial artificial intelligence. *IEEE Transactions on Industrial Informatics*, 16(10):6532–6542, 2019.

[41] Borja Balle and Yu-Xiang Wang. Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In *International Conference on Machine Learning*, pages 394–403. PMLR, 2018.

[42] Jiahao Ding, Guannan Liang, Jinbo Bi, and Miao Pan. Differentially private and communication efficient collaborative learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7219–7227, 2021.

[43] Zhize Li and Peter Richtárik. Canita: Faster rates for distributed convex optimization with communication compression. *Advances in Neural Information Processing Systems*, 34:13770–13781, 2021.

[44] Simon Lacoste-Julien, Mark Schmidt, and Francis Bach. A simpler approach to obtaining an $O(1/t)$ convergence rate for the projected stochastic subgradient method. *arXiv preprint arXiv:1212.2002*, 2012.

[45] Sören Sonnenburg, Gunnar Rätsch, Christin Schäfer, and Bernhard Schölkopf. Large scale multiple kernel learning. *The Journal of Machine Learning Research*, 7:1531–1565, 2006.

[46] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[47] Andrew Lowy and Meisam Razaviyayn. Private federated learning without a trusted server: Optimal algorithms for convex losses, 2023.

[48] Thomas Steinke. Composition of differential privacy & privacy amplification by subsampling. *arXiv preprint arXiv:2210.00597*, 2022.

VIII. Appendix

A. Supporting Lemmas

Lemma 4. *[27] Let $b_0 > 0$ and define the sequence $a_{t+1} \leq b_0 + \frac{1}{2}a_t^{\frac{3}{2}}$ where $a_0 \leq \frac{16}{9}$. Then, after $T = \Theta(\log\log(\frac{1}{b}))$, we have $a_T = O(b_0)$.*

The following lemma describes properties of $\phi_i(v; w)$ [27].

Lemma 5. *For any $i \in \{1, 2, \dots, n\}$, ϕ_i defined in (8) satisfies the following properties:*

1) *For every $M \geq 0$ and $w, v \in \mathcal{X}$ such that $v \neq w$,*
$$\begin{aligned}&\nabla_v^2 \phi_i(v; w)\\ =&\nabla^2 f_i(w) + \frac{M}{2}\|v - w\| I_d + \frac{M}{2\|v - w\|}(v - w)(v - w)^T.\end{aligned}$$
Therefore, $\nabla_v^2 \phi_i(v; w) \succeq \lambda_{\min}(\nabla^2 f_i(w)) I_d + M\|v - w\| I_d$.
2) *For every $M \geq L_2$, and $v, w \in \mathcal{X}$,*
$$f_i(v) \leq \phi_i(v; w).$$
3) *For every $M \geq 0$ and $v, w \in \mathcal{X}$,*
$$\phi_i(v; w) \leq f_i(v) + \frac{M + L_2}{6}\|v - w\|^3.$$

It can be verified that $\phi(v; w) = \frac{1}{n}\sum_{i=1}^n \phi_i(v; w)$. Therefore, we can obtain similar properties between ϕ and f as in Lemma 5.

B. Proof of Theorem 1

We begin by introducing some relevant properties of DP.

Lemma 6. *(Privacy for Subsampling [48]) Suppose $\mathcal{G}$ is an (ε, δ)-DP mechanism. Let $\texttt{Sample}_{r_1,r_2} : \mathcal{D}^{r_1} \to \mathcal{D}^{r_2}$ be the subsampling operation that takes a dataset belonging to $\mathcal{D}^{r_1}$ as input and selects uniformly at random a subset of $r_2 \leq r_1$ elements from the input dataset. Consider the mechanism*
$$\mathcal{G} \circ \texttt{Sample}_{r_1,r_2}(D),$$
where $D \in \mathcal{D}^{r_1}$. Then the mechanism $\mathcal{G} \circ \texttt{Sample}_{r_1,r_2}$ is (ε', δ')-DP for $\varepsilon' = \log(1 + r_2(e^\varepsilon - 1)/r_1)$ and $\delta' = r_2\delta/r_1$.

Lemma 7. *(Composition of DP [48]) Given T randomized algorithms $\mathcal{A}_1, \dots, \mathcal{A}_T$, each of which is $(\varepsilon_i, \delta_i)$-DP with $\varepsilon_i \in (0, 0.9]$ and $\delta_i \in (0, 1]$. Then $\mathcal{A}$ with $\mathcal{A}(\cdot) = (\mathcal{A}_1(\cdot), \dots, \mathcal{A}_T(\cdot))$ is $(\tilde{\varepsilon}, \tilde{\delta})$-DP with*
$$\tilde{\varepsilon} = \sqrt{\sum_{t=1}^T 2\varepsilon_t^2 \log\left(e + \frac{\sqrt{\sum_{t=1}^T \varepsilon_t^2}}{\hat{\delta}}\right) + \sum_{t=1}^T \varepsilon_t^2}$$
and
$$\tilde{\delta} = 1 - (1 - \hat{\delta})\prod_{t=1}^T(1 - \delta_t)$$
for any $\hat{\delta} \in (0, 1]$.

DP at each local computation: The noise added to each coordinate of $[\text{grad}_i^{t,s}]_{c_i^t}$ is drawn from the Gaussian distribution $\mathcal{N}(0, \sigma^2)$. Then based on Lemma 1, each local iteration of Algorithm 2 achieves $(\varepsilon_s, \delta_0)$-DP for each sampled data $\zeta_{i,t}$ with
$$\varepsilon_s = \frac{2\sqrt{k\log(1.25/\delta_0)}(G_1 + G_2 D)}{\sigma\sqrt{d}}$$
for any $\delta_0 \in [0, 1]$.

By the sub-sampling amplification property of DP in Lemma 6, each iteration of Algorithm 2 achieve $(\varepsilon_s', \delta_0/m)$-DP for client i's local dataset ζ_i, where
$$\varepsilon_s' = \log\left(1 + \frac{e^{\varepsilon_s} - 1}{m}\right) \leq \frac{2\varepsilon_s}{m}.$$
Due to the conditions on σ and T, we obtain
$$\varepsilon_s'^2 \leq \frac{16k\log(1.25/\delta_0)(L_0 + L_1 D)^2}{\sigma^2 m^2 d} \leq \frac{\varepsilon^2}{5\tau T} \leq 0.8.$$
Therefore, we have $\varepsilon_s \leq 0.9$ and
$$\sum_{s=1}^{\tau T} \varepsilon_s^2 \leq \frac{1}{5}\sum_{s=1}^{\tau T} \varepsilon^2 \leq 1$$
for the given $\varepsilon \in (0, 1]$.

DP after T iterations: After T communication rounds, client i performs $T\tau$ iterations of local computation. By Lemma 7, we obtain Algorithm 1 is $(\tilde{\varepsilon}, \tilde{\delta})$-DP with
$$\tilde{\varepsilon} = \sqrt{\sum_{s=1}^{\tau T} 2\varepsilon_s^2 \log\left(e + \frac{\sqrt{\sum_{s=1}^{\tau T} \varepsilon_s^2}}{\tilde{\delta}}\right) + \sum_{s=1}^{\tau T} \varepsilon_s^2}$$
and $\tilde{\delta} = 1 - (1 - \delta')(1 - \delta_0/m)^{\tau T}$ for any $\delta' \in (0, 1]$. Furthermore, there holds
$$\begin{aligned}\tilde{\varepsilon} =& \sqrt{\sum_{s=1}^{\tau T} 2\varepsilon_s^2 \log\left(e + \frac{\sqrt{\sum_{s=1}^{\tau T} \varepsilon_s^2}}{\tilde{\delta}}\right) + \frac{1}{5}\varepsilon^2}\\ \leq& \sqrt{3\sum_{s=1}^{\tau T} \varepsilon_s^2 + \frac{1}{5}\varepsilon}\\ =& \sqrt{\frac{3}{5}\varepsilon^2 + \frac{1}{5}\varepsilon}\\ \leq& \varepsilon,\end{aligned}$$
where we fix $\delta' = \sqrt{\sum_{s=1}^{\tau T} \varepsilon_s^2}$. By setting $\delta = \tilde{\delta}$, we have Algorithm 1 is (ε, δ)-DP.

C. Proof of Lemma 3

We can write a stochastic estimate of $\phi_i(v; w)$ as follows:
$$\begin{aligned}&\hat{\phi}_i(v; w)\\ \triangleq& f(w) + \langle \hat{g}_i, v - w\rangle + \frac{1}{2}\left\langle \hat{H}_i(v - w), v - w\right\rangle + \frac{M}{6}\|v - w\|^3,\end{aligned}$$
where $\hat{g}_i$ and $\hat{H}_{i,t}$ are stochastic estimates of $\nabla f_i(w)$ and $\nabla^2 f_i(w)$. According to Algorithm 1, we find that grad_s is

a stochastic gradient of $\nabla_{\theta_s}\phi(\theta_s,\theta_0)$. Based on the non-expansive property of the projection operator, we have

$$\begin{aligned}&\mathbb{E}\left[\|\theta_{s+1}-\theta_*\|^2|\mathcal{F}_s\right]\\ \le&\|\theta_s-\theta_*\|^2+\eta_s^2\mathbb{E}\left[\|\mathrm{grad}_s+b_s\|^2|\mathcal{F}_s\right]\\ &-2\eta_s\left\langle\nabla_{\theta_s}\phi_i(\theta_s;\theta_0),\theta_s-\theta_0\right\rangle\\ \le&\|\theta_s-\theta_*\|^2+\eta_s^2\mathbb{E}\left[\|\mathrm{grad}_s+b_s\|^2|\mathcal{F}_s\right]\\ &-2\eta_s\left[\phi_i(\theta_s;\theta_0)-\phi_i(\theta^*;\theta_0)+\frac{\mu}{2}\|\theta_s;\theta_0\|^2\right],\end{aligned}$$

where the last inequality holds from the μ-strong convexity of $\nabla\phi_i$. By arranging the inequality, we have

$$\begin{aligned}&\mathbb{E}[\phi_i(\theta_{s+1};\theta_0)]-\phi_i(\theta^*;\theta_0)\\ \le&\frac{\eta_s(L^2+\sigma^2d)}{2}+\left(\frac{1}{2\eta_s}-\frac{\mu}{2}\right)\mathbb{E}[\|\theta_s-\theta^*\|^2]\\ &-\frac{1}{2\eta_s}\mathbb{E}[\|\theta_{s+1}-\theta^*\|^2],\end{aligned}\tag{11}$$

where $L=L_0+L_1D+\frac{M}{2}D^2$. With $\eta_s=\frac{2}{\mu(s+2)}$ and multiplying the (11) by $s+1$, we obtain

$$\begin{aligned}&(s+1)[\mathbb{E}[\phi_i(\theta_{s+1};\theta_0)]-\phi_i(\theta^*;\theta_0)]\\ \le&\frac{(s+1)(L^2+\sigma^2d)}{\mu(s+2)}-\frac{\mu(s+2)(s+1)}{4}\mathbb{E}[\|\theta_{s+1}-\theta^*\|^2]\\ &+\left(\frac{\mu(s+2)(s+1)}{4}-\frac{\mu(s+1)}{2}\right)\mathbb{E}[\|\theta_s-\theta^*\|^2]\\ \le&\frac{L^2+\sigma^2d}{\mu}+\frac{\mu}{4}\Big[s(s+1)\mathbb{E}\left[\|\theta_s-\theta^*\|^2\right]\\ &\qquad-(s+1)(s+2)\mathbb{E}\left[\|\theta_{s+1}-\theta^*\|^2\right]\Big]\end{aligned}$$

By summing from $s=0$ to $s=\tau$ of these s-weighted inequalities, we have

$$\begin{aligned}&\sum_{s=0}^{\tau-1}(s+1)[\mathbb{E}[\phi_i(\theta_{s+1};\theta_0)]-\phi_i(\theta^*;\theta_0)]\\ \le&\frac{\tau(L^2+\sigma^2d)}{\mu}-\frac{\mu}{4}\tau(\tau+1)\mathbb{E}\left[\|\theta_\tau-\theta^*\|^2\right].\end{aligned}$$

Thus,

$$\mathbb{E}\phi_i\left(\frac{2}{\tau(\tau+1)}\sum_{s=0}^{\tau-1}(s+1)\theta_s;\theta_0\right)-\phi_i(\theta^*;\theta_0)\le\frac{2(L^2+\sigma^2d)}{\mu(\tau+1)}.$$

Therefore, after the local computation of Algorithm 2, the suboptimality gap is given by

$$O\left(\frac{L^2+\sigma^2d}{\mu\tau}\right).$$

Then, we plug in the value of σ in (7) to obtain the suboptimality gap:

$$O\left(\left(\frac{L^2}{\mu\tau}+\frac{k\log(1/\delta_0)(G_1+G_2D)^2}{\varepsilon^2m^2\mu}\right)\right).$$

Then, by setting the number of local iterations to $\tau=\frac{L^2\varepsilon^2m^2}{Tk\log(1/\delta_0)}$, we obtain that the subotimality is given by (10).

D. Proof of Theorem 2

Using 2) in Lemma 5, we can write

$$\begin{aligned}&\mathbb{E}[f(x_{t+1})]-f(x^*)\\ \le&\mathbb{E}[\phi(x_{t+1};x_t)]-f(x^*)\\ =&\mathbb{E}\left[\phi(x_{t+1};x_t)-\frac{1}{n}\sum_{i=1}^n\phi_i(x_{i,t+1};x_t)+\frac{1}{n}\sum_{i=1}^n\phi_i(x_{i,t+1};x_t)\right]\\ &-\frac{1}{n}\sum_{i=1}^n\min_{x^{(i)}\in\mathcal{X}}\phi_i(x^{(i)};x_t)+\frac{1}{n}\sum_{i=1}^n\min_{x^{(i)}\in\mathcal{X}}\phi_i(x^{(i)};x_t)\\ &-f(x^*)\\ \le&\frac{1}{n}\sum_{i=1}^n\left[\mathbb{E}[\phi_i(x_{i,t+1};x_t)]-\min_{x^{(i)}\in\mathcal{X}}\phi_i(x^{(i)};x_t)\right]\\ &+\mathbb{E}\left[\phi(x_{t+1};x_t)-\frac{1}{n}\sum_{i=1}^n\phi_i(x_{i,t+1};x_t)\right]\\ &+\left[\min_{x\in\mathcal{X}}\phi(x;x_t)-f(x^*)\right]\end{aligned}\tag{12}$$

where the last inequality uses the fact that $\frac{1}{n}\sum_{i=1}^n\min_{x^{(i)}\in\mathcal{X}}\phi_i(x;x_t)\le\min_{x\in\mathcal{X}}\phi(x;x_t)$. Since $\phi(x;x_t)$ is a strongly convex function of x and $\mathcal{X}$ is a closed and convex set, there exists a unique $x^*_{t+1}=\arg\min_{x\in\mathcal{X}}\phi(x;x_t)$.

$$\begin{aligned}&\frac{1}{n}\sum_{i=1}^n\left[\mathbb{E}[\phi_i(x_{i,t+1};x_t)]-\min_{x^{(i)}\in\mathcal{X}}\phi_i(x^{(i)};x_t)\right]\\ \le&O\left(\frac{k\log(1/\delta_0)(G_1+G_2D)^2T}{\varepsilon^2m^2\mu}\right)\triangleq\Gamma_1\end{aligned}$$

In each step of Algorithm 1, we find an approximate minimizer of $\phi_i(x;x_t)$ using the solver in Algorithm 2. Lemma 3 provides the performance guarantee of the GMSolver and shows that at each step of Algorithm 1, the optimization error in minimizing $\phi_i(x^{(i)};x_i)$ is less than Γ_1.

For the second term of (12), we obtain

$$\begin{aligned}&\mathbb{E}\left[\phi(x_{t+1};x_t)-\frac{1}{n}\sum_{i=1}^n\phi_i(x_{i,t+1};x_t)\right]\\ =&\frac{1}{n}\sum_{i=1}^n\mathbb{E}\left[\phi_i(x_{t+1};x_t)-\phi_i(x_{i,t+1};x_t)\right]\\ \le&\frac{1}{n}\sum_{i=1}^n\mathbb{E}[\nabla_{x_{t+1}}\phi_i(x_{t+1};x_t)(x_{t+1}-x_{i,t+1})]\\ =&\frac{1}{n}\sum_{i=1}^n\nabla_{x_{t+1}}\phi_i(x_{t+1};x_t)\mathbb{E}[(x_{t+1}-x_t)-(x_{i,t+1}-x_t)]\\ =&\frac{1}{n}\sum_{i=1}^n\nabla_{x_{t+1}}\phi_i(x_{t+1};x_t)\mathbb{E}\left[\frac{\alpha}{n}\sum_{i=1}^n\mathcal{S}_i^t(y_{i,t})-\alpha y_{i,t}\right]\\ =&\frac{1}{n}\sum_{i=1}^n\nabla_{x_{t+1}}\phi_i(x_{t+1};x_t)\mathbb{E}\left[\frac{\alpha}{n}\sum_{i=1}^n y_{i,t}-\alpha y_{i,t}\right]\\ \le&\alpha LD\triangleq\alpha\Gamma_2,\end{aligned}\tag{13}$$

where the first inequality follows from the Lipschitz continuous of ϕ_i, and the third equality holds from the steps 8 and 9 in Algorithm 1. Putting

$$\alpha = \frac{\Gamma_1}{LD}$$

into (13), we have

$$\phi(x_{t+1}; x_t) - \frac{1}{n}\sum_{i=1}^{n} \phi_i(x_{i,t+1}; x_t) \leq O(\Gamma_1)$$

Then, we provide an upper bound on the last term of (12). We obtain the following relationship by 3) in Lemma 5.

$$\begin{aligned}&\min_{x\in\mathcal{X}} \phi(x; x_t) - f(x^*)\\ \leq &\min_{x\in\mathcal{X}} \left[f(x) + \frac{M+L_2}{6}\|x - x_t\|^3 - f(x^*)\right].\end{aligned}$$

Since $\mathcal{X}$ is a convex set and $x_t, x^* \in \mathcal{X}$, for all $\eta \in [0,1]$, $(1-\eta)x_t + \eta x^* \in \mathcal{X}$. Therefore,

$$\begin{aligned}&\min_{x\in\mathcal{X}} \left[f(x_t) + \frac{M+L_2}{6}\|x - x_t\|^3 - f(x^*)\right]\\ \leq &\min_{\eta_t\in[0,1]} \left[f\big((1-\eta_t)x_t + \eta_t x^*\big) + \eta_t^3 \frac{M+L_2}{6}\|x_t - x^*\|^3 \right.\\ &\left. - f(x^*)\right].\end{aligned}$$

By the convexity of f, we have $f((1-\eta_t)x_t+\eta_t x^*) \leq f(x_t) - f(x^*) - \eta_t\left(f(x_t) - f(x^*)\right)$. Also, strong convexity implies that $\|x_{t+1} - x^*\|^3 \leq \left[\frac{2}{\mu}(f(x_t) - f(x^*)\right]^{\frac{3}{2}}$. Thus,

$$\begin{aligned}&\min_{x\in\mathcal{X}} \phi(x; x_t) - f(x^*)\\ \leq &\min_{\eta_t\in[0,1]} \left\{ f(x_t) - f(x^*) - \eta_t(f(x_t) - f(x^*)) \right.\\ &\left. + \eta_t^3 \frac{M+L_2}{6}\left[\frac{2}{\mu}(f(x_t) - f(x^*)\right]^{\frac{3}{2}} \right\} \end{aligned} \tag{14}$$

Let $\lambda = \left(\frac{3}{M+L_2}\right)^2\left(\frac{\mu}{2}\right)^3$ and $q_t = \lambda^{-1}\big(f(x_t) - f(x^*)\big)$. Based on (14), we can rephrase (12) as

$$q_{t+1} \leq \lambda^{-1}\Gamma_1 + \min_{\eta_t\in[0,1]} \left(q_t - \eta_t q_t + \frac{1}{2}\eta_t^3 q_t^{\frac{3}{2}}\right). \tag{15}$$

Denote $\eta_t^* = \arg\min_{\eta_t\in[0,1]} \left(q_t - \eta_t q_t + \frac{1}{2}\eta_t^3 q_t^{\frac{3}{2}}\right)$, we have that $\eta_t = \min\left\{\sqrt{\frac{2}{3\sqrt{q_t}}}, 1\right\}$.

Phase I: If $q_t \geq \frac{4}{9}$, then $\eta^* = \sqrt{\frac{2}{3\sqrt{q_t}}}$. The iteration (15) will become

$$q_{t+1} \leq \lambda^{-1}\Gamma_1 + q_t - \left(\frac{2}{3}\right)^{\frac{3}{2}} q_t^{\frac{3}{4}}.$$

Phase II: If $q_t < \frac{4}{9}$, then $\eta^* = 1$. The iteration (15) will be given by

$$q_{t+1} \leq \lambda^{-1}\Gamma_1 + \frac{1}{2} q_t^{\frac{3}{2}}.$$

Assume that $q_0 \geq \frac{4}{9}$. We will show that, $\{q_t\}_{t\in[T]}$ is a decreasing sequence, and as a result there exists $T_1 > 0$, such that $q_t < \frac{4}{9}$ for $t \geq T_1$, and as a result, $\eta_t^* = 1$ for $t \geq T_1$.

Convergence of Phase I: To prove the convergence of Phase I, we follow the techniques in Nesterov and Polyak [37]. Let $\tilde{q}_{t+1} = \frac{9}{4} q_t$, and assume $\Gamma_1 \leq \frac{4\lambda}{27}$. Then, we can rephrase the recursion for Phase I as follows:

$$\begin{aligned}\tilde{q}_{t+1} \leq &\frac{9\Gamma_1}{4\lambda} + \tilde{q}_t - \frac{2}{3}\tilde{q}_t^{\frac{3}{4}}\\ \leq &\tilde{q}_t - \frac{1}{3}\tilde{q}_t^{\frac{3}{4}},\end{aligned}$$

where the last step follows from $\tilde{q}_t \geq 1$ and $\frac{9\Gamma_1}{4\lambda} \leq \frac{1}{3} \leq \frac{\tilde{q}_t^{\frac{3}{4}}}{3}$. It also shows that provided that $\tilde{q}_t \geq 1$, $\tilde{q}_{t+1} \leq \tilde{q}_t$.

Using induction, it is straightforward to show that in Phase I

$$\frac{9\tilde{q}_{t+1}}{4} \leq \left[\left(\frac{9\tilde{q}_t}{4}\right)^{\frac{1}{4}} - \frac{t}{12}\right]^4.$$

This result implies that after T_1^* iterations where

$$T_1^* \leq O\left(\frac{\sqrt{M+L_2}(f(x_0) - f(x^*))^{\frac{1}{4}}}{\mu^{\frac{3}{4}}}\right), \tag{16}$$

we have $q_{T_1^*} < \frac{4}{9}$, and we enter Phase II.

Convergence of Phase II: The recursion in Phase II is given by

$$q_{t+1} \leq \lambda^{-1}\Gamma_1 + \frac{1}{2} q_t^{\frac{3}{2}}.$$

We define another sequence $\{r_t\}_{t\geq 0}$ as follows:

$$r_0 = q_0,\ r_{t+1} = \frac{3}{4}(r_t)^{\frac{3}{2}}.$$

By induction, we can easily prove that for every $t \geq 0$ such that $\lambda^{-1}\Gamma_1 \leq \frac{1}{4} r_t^{\frac{3}{2}}$, we have $r_{t+1} \geq q_{t+1}$. Then, we can write

$$r_{t+1} = \frac{3}{4} r_t^{\frac{3}{2}} \Leftrightarrow \frac{9}{16} r_{t+1} = \left(\frac{9}{16} r_t\right)^{\frac{3}{2}}.$$

Therefore, we obtain that $\log(\frac{9}{16} r_t) = (\frac{3}{2})^t \log(\frac{9}{16} r_0)$. We want to find T such that $\lambda^{-1}\Gamma_1 \leq \frac{1}{4} r_T^{\frac{3}{2}} \leq 2\lambda^{-1}\Gamma_1$ which is equivalent to $\frac{2}{3}\log(\frac{27}{16}\lambda^{-1}\Gamma_1) \leq \log(\frac{9}{16} r_T) \leq \frac{2}{3}\log(\frac{27}{8}\lambda^{-1}\Gamma_1)$. Then, we can see that $T = \Theta\left(\log\left(\log\left(\frac{\lambda}{\Gamma_1}\right)\right)\right)$. Therefore, we have $r_{T+1} = O(\lambda^{-1}\Gamma_1)$. Also, by the construction, $q_{T+1} \leq r_{T+1} = O(\lambda^{-1}\Gamma_1)$. Therefore, the number of iterations to achieve the minimum optimization error in Phase II is

$$T_2^* = \tilde{\Theta}\left(\log\log\left(\frac{\varepsilon m}{\sqrt{k\log(1/\delta_0)}}\right)\right). \tag{17}$$

Finally, the optimization error is given by

$$\begin{aligned}&f(x_T) - f(x^*)\\ =&O\left(\frac{k\log(1/\delta_0)(G_1 + G_2 D)^2}{\varepsilon^2 m^2 \mu} \cdot (T_1^* + T_2^*)\right),\end{aligned}$$

where $T = T_1^* + T_2^*$ and T_1^* and T_2^* are given by (16) and (17), respectively.