Federated Learning With Sparsified Model Perturbation: Improving Accuracy Under Client-Level Differential Privacy

Rui Hu, Member, IEEE, Yuanxiong Guo, Senior Member, IEEE, and Yanmin Gong, Senior Member, IEEE

Abstract—Federated learning (FL) that enables edge devices to collaboratively learn a shared model while keeping their training data locally has received great attention recently and can protect privacy in comparison with the traditional centralized learning paradigm. However, sensitive information about the training data can still be inferred from model parameters shared in FL. Differential privacy (DP) is the state-of-the-art technique to defend against those attacks. The key challenge to achieving DP in FL lies in the adverse impact of DP noise on model accuracy, particularly for deep learning models with large numbers of parameters. This paper develops a novel differentially-private FL scheme named Fed-SMP that provides a client-level DP guarantee while maintaining high model accuracy. To mitigate the impact of privacy protection on model accuracy, Fed-SMP leverages a new technique called Sparsified Model Perturbation (SMP) where local models are sparsified first before being perturbed by Gaussian noise. We provide a tight end-to-end privacy analysis for Fed-SMP using Rényi DP and prove the convergence of Fed-SMP with both unbiased and biased sparsifications. Extensive experiments on real-world datasets are conducted to demonstrate the effectiveness of Fed-SMP in improving model accuracy with the same DP guarantee and saving communication cost simultaneously.

Index Terms—Federated learning, edge computing, differential privacy, communication efficiency, sparsification.

I. INTRODUCTION

The proliferation of edge devices such as smartphones and Internet-of-things (IoT) devices, each equipped with rich sensing, computation, and storage resources, leads to tremendous data being generated on a daily basis at the network edge. These data can be analyzed to build machine learning models that enable a wide range of intelligent services such as personal fitness tracking [1], traffic monitoring [2], and smart home security [3]. Traditional machine learning paradigm requires transferring all the raw data to the cloud before training a model, leading to high communication cost and severe privacy risk.

As an alternative machine learning paradigm, Federated Learning (FL) has attracted significant attention recently due to its benefits in communication efficiency and privacy [4]. In FL, multiple clients collaboratively learn a shared statistical model under the orchestration of the cloud without sharing their local data [5]. Although only model updates instead of raw data are shared by each client in FL, it is not sufficient to ensure privacy as the sensitive training data can still be inferred from the shared model parameters by using advanced inference attacks. For instance, given an input sample and a target model, the membership inference attack [6] can train an attack model to determine whether the sample was used for training the target model or not. Also, given a target model and class, the model inversion attack [7] can recover the typical representations of the target class using an inversion model learned from the correlation between the inputs and outputs of the target model.

As a cryptography-inspired rigorous definition of privacy, differential privacy (DP) has become the de-facto standard for achieving data privacy and can give a strong privacy guarantee against an adversary with arbitrary auxiliary information [8]. For privacy protection in FL, client-level DP is often more relevant than record-level DP: client-level DP guarantee protects the entire local dataset of a client, while record-level DP guarantees protect only a single data sample of a client [9], [10]. While DP can be straightforwardly achieved using Gaussian or Laplacian mechanism [8], achieving client-level DP in the FL setting faces several major challenges in maintaining high model accuracy. First, FL is an iterative learning process where model updates are exchanged in multiple rounds, leading to more privacy leakage compared to the one-shot inference. Second, the intensity of added DP noise is linearly proportional to the model size, which can be very large (e.g., millions of model parameters) for modern deep neural networks (DNNs), and will severely degrade the accuracy of the trained model. Third, it is more challenging to achieve client-level DP than record-level DP because the entire dataset of a client rather than a single data sample of that client needs to be protected. Existing studies [11], [12], [13] on FL with client-level DP suffer from significant...
accuracy degradation due to the inherent challenge of large additive random noise required to achieve a certain level of client-level DP for DNNs. While some works \cite{14,15,16} rely on specialized techniques like model sparsification to reduce the impact of DP noise, these methods are limited to record-level DP and often lack a comprehensive theoretical analysis for their effectiveness.

In this paper, we propose a new differentially-private FL scheme called Fed-SMP, which guarantees client-level DP while preserving model accuracy and saving communication cost simultaneously. Fed-SMP utilizes Sparsified Model Perturbation (SMP) to improve the privacy-accuracy tradeoff of DP in FL. Specifically, SMP first sparsifies the local model update of a client at each round by selecting only a subset of coordinates to keep and then adds Gaussian noise to perturb the values at those selected coordinates. As we will show later in this paper, by using SMP in FL, the sparsification will have an amplification effect on the privacy guarantee offered by the added Gaussian noise, leading to a better privacy-accuracy tradeoff. Meanwhile, it can reduce the communication cost by compressing the shared model updates at each round. In summary, the main contributions of this paper are summarized as follows.

- We propose a new differentially private FL scheme called Fed-SMP, which achieves client-level DP guarantee for FL with a large number of clients. Fed-SMP only makes lightweight modifications to FedAvg \cite{5}, the most common learning method for FL, which enables easy integration into existing packages.

- Compared with DP-FedAvg \cite{11}, the state-of-art of differentially private FL schemes, Fed-SMP improves the privacy-utility tradeoff and uplink communication efficiency for FL with client-level DP, by using sparsification as a tool for amplifying privacy and reducing the uplink communication cost at the same time.

- By integrating different sparsification operators with model perturbation, we design two algorithms of Fed-SMP: Fed-SMP with unbiased random sparsification and Fed-SMP with biased top-k sparsification. The resulting algorithms require less amount of added random noise to achieve the same level of DP and simultaneously are compatible with secure aggregation, a crucial privacy-enhancing technique to achieve client-level DP in practical FL systems, to reduce the size of the model to be transmitted.

- To prove the \((\epsilon, \delta)\)-DP guarantee of Fed-SMP, we use Rényi differential privacy (RDP) to tightly account the end-to-end privacy loss of Fed-SMP. We also theoretically analyze the impact of sparsified model perturbation on the convergence of Fed-SMP with both unbiased and biased sparsifications considering non-IID data distribution, filling the gap in the state-of-arts. The theoretical results indicate that under a certain DP guarantee, the optimal compression level of Fed-SMP needs to balance the increased compression error and reduced privacy error due to sparsification.

- We empirically evaluate the performances of Fed-SMP on both IID and non-IID datasets and compare the results with those of the state-of-art baselines. Experimental results demonstrate that Fed-SMP can achieve higher model accuracy than baseline approaches under the same level of DP and save the communication cost simultaneously.

The rest of the paper is organized as follows. Preliminaries on DP are described in Section II. Section III introduces the problem formulation and presents the proposed Fed-SMP scheme. The privacy and convergence properties of Fed-SMP with random and top-\(k\) sparsification strategies are rigorously analyzed in Section V. Section VI shows the experimental results. Finally, Section VII reviews the related work, and Section VIII concludes the paper.

II. PRELIMINARIES

DP has been proposed as a rigorous privacy notion for measuring privacy risk. The classic notion of DP, \((\epsilon, \delta)\)-DP, is defined as follows:

\[
\text{Definition 1 (}\ ((\epsilon, \delta)\)-DP \cite{8}) : \text{ Given privacy parameters } \epsilon > 0 \text{ and } 0 \leq \delta < 1, \text{ a randomized mechanism } M \text{ satisfies } (\epsilon, \delta)-\text{DP if for any two adjacent datasets } D, D' \text{ and any subset of outputs } O \subseteq \text{range}(M), \\
\text{Pr}[M(D) \in O] \leq e^\epsilon \text{Pr}[M(D') \in O] + \delta. \quad (1)
\]

When \(\delta = 0\), we have \(\epsilon\)-DP, or Pure DP.

To better calculate the privacy loss over multiple iterations in differentially private learning algorithms, Rényi differential privacy (RDP) has been proposed as follows:

\[
\text{Definition 2 ((}\alpha, \rho\text{)-RDP \cite{17}) : \text{ Given a real number } \alpha > 1 \text{ and privacy parameter } \rho \geq 0, \text{ a randomized mechanism } M \text{ satisfies } (\alpha, \rho)\text{-RDP if for any two adjacent datasets } D, D', \text{ the Rényi } \alpha\text{-divergence between } M(D) \text{ and } M(D') \text{ satisfies} \\
D_{\alpha}[M(D)∥M(D')] := \frac{1}{\alpha - 1} \log \mathbb{E} \left[ \left( \frac{M(D)}{M(D')} \right)^{\alpha} \right] \leq \rho(\alpha).
\]

RDP is a relaxed version of pure DP with a tighter composition bound. Thus, it is more suitable to analyze the end-to-end privacy loss of iterative algorithms. We can convert RDP to \((\epsilon, \delta)\)-DP for any \(\delta > 0\) using the following lemma:

\[
\text{Lemma 1 (From RDP to } (\epsilon, \delta)\text{-DP \cite{18}) : \text{ If the randomized mechanism } M \text{ satisfies } (\alpha, \rho(\alpha))\text{-RDP, then it also satisfies} \\
(\rho(\alpha) + \frac{\log(1/\delta)}{\alpha-1}, \delta)\text{-DP.}
\]

In the following, we provide some useful definitions and lemmas about DP and RDP that will be used to derive our main results in the rest of the paper.

\[
\text{Definition 3 (} \ell_2\text{-sensitivity \cite{8}) : \text{ Let } h : D \rightarrow \mathbb{R}^d \text{ be a query function over a dataset. The } \ell_2\text{-sensitivity of } h \text{ is defined as } } \psi(h) := \sup_{D, D' \in D, D \neq D'} \|h(D) - h(D')\|_2 \text{ where } D \sim D' \text{ denotes that } D \text{ and } D' \text{ are two adjacent datasets.}
\]

\[
\text{Lemma 2 (Gaussian Mechanism \cite{17}) : \text{ Let } h : D \rightarrow \mathbb{R}^d \text{ be a query function with } \ell_2\text{-sensitivity } \psi(h). \text{ The Gaussian mechanism } M = h(D) + N(0, \sigma^2 \psi(h)^2 I_d) \text{ satisfies } (\alpha, \alpha/2\sigma^2)\text{-RDP.}
\]

\[
\text{Lemma 3 (RDP for Subsampling Mechanism \cite{18}, \cite{19}) : \text{ For a Gaussian mechanism } M \text{ and any m-datasets dataset } D, \text{ define } M \circ SUBSAMPLE \text{ as 1) subsample without replacement } B \text{ datapoints from the dataset (denote } q = B/m \text{ as the sampling ratio); and 2) apply } M \text{ on the subsampled dataset as input. Then if } M \text{ satisfies } (\alpha, \rho(\alpha))\text{-RDP with respect to the subsampled dataset for all integers } \alpha \geq 2, \text{ then}
\]
Furthermore, some clients can collude with the server or each other to infer private information about a specific victim client. Besides, the adversary could also be the passive outside attacker. These attackers can eavesdrop on all the shared messages during the execution of the training but will not actively inject false messages into or interrupt message transmissions.

C. Achieving Client-Level DP in FL

As the classic and most widely-used algorithm in the FL setting, Federated Averaging (FedAvg) [5] solves (2) by running multiple iterations of stochastic gradient descent (SGD) in parallel on a subset of clients and then averaging the resulting local model updates at a central server periodically. Compared with distributed SGD, FedAvg is shown to achieve the same model accuracy with fewer communication rounds. Specifically, FedAvg involves $T$ communication rounds, and each round can be divided into four stages as shown in Fig. 1. First, at the beginning of round $t \in \{0, \ldots, T-1\}$, the server randomly selects a subset $\mathcal{W}^t$ of $r$ clients and sends them the latest global model $\theta^n$ to perform local computations. Second, each client $i \in \mathcal{W}^t$ runs $\tau$ iterations of SGD on its local dataset to update its local model. Let $\theta^{t,s}_i$ denote client $i$’s model after the $s$th local iteration at the $t$th communication round where $s \in [0, \tau-1]$. By model initialization, we have $\theta^{t,0}_i = \theta^t$. Then the update rule at client $i$ is represented as

$$
\theta^{t,s+1}_i = \theta^{t,s}_i - \eta_i \nabla_{\theta^{t,s}_i} f_i(\theta^{t,s}_i), \forall s = 0, \ldots, \tau - 1,
$$

(3)

where $\eta_i$ is the local learning rate and $\nabla_{\theta^{t,s}_i} f_i(\theta^{t,s}_i, z)$ represents the stochastic gradient over a mini-batch $\xi^{t,s}_i$ of $B$ datapoints sampled from $D_i$. Third, the client $i$ uploads its model update $\Delta^t_i := \theta^{t,s}_i - \theta^{t,s-1}_i$ to the server. Fourth, after receiving all the local model updates, the server updates the global model by

$$
\theta^{t+1} = \theta^t - \frac{1}{r} \sum_{i \in \mathcal{W}^t} \Delta^t_i.
$$

(4)

The same procedure repeats at the next round $t+1$ thereafter until satisfying certain convergence criteria.

In FedAvg, clients repeatedly upload local model updates of large dimension $d$ (e.g., millions of model parameters for DNNs) to the server and download the newly-updated global model from the server many times to learn an accurate global model. Since the bandwidth between the server and clients could be rather limited, especially for uplink transmissions, the overall communication cost could be very high. Furthermore, although FedAvg can avoid direct information leakage by keeping the local dataset on the client, the model updates shared at each round can still potentially leak privacy. Adversaries in FL (as defined in Section III-B) can infer/reconstruct private information of clients’ local training data from these shared local model updates, using recent advanced attacks such as membership inference attack [6] and model inversion attack [7].

As the commonly-used privacy protection technique for machine learning, DP mechanisms have been used to mitigate the above-mentioned privacy risks in FedAvg and provide rigorous privacy guarantees. In FL, DP can be categorized into record-level DP and client-level DP, depending on how the adjacent
Algorithm 1: DP-FedAvg.

Require: model dimension \( d \), number of clients \( n \), number of selected clients per round \( r \), number of training rounds \( T \), local update period \( \tau \), local learning rate \( \eta_l \), clipping threshold \( C \), noise multiplier \( \sigma \).

Server executes:
1: Initialize \( \theta^0 \in \mathbb{R}^d \)
2: for \( t = 0 \) to \( T-1 \) do
3: Sample a set of \( r \) clients uniformly at random without replacement, denoted by \( V_t \subseteq [n] \)
4: Broadcast \( \theta^t \) to all clients in \( V_t \)
5: for each clients \( i \in \mathcal{W}^t \) in parallel do
6: \( y^t_i \leftarrow \text{ClientUpdate}(\theta^t) \)
7: end for
8: \( \theta^{t+1} \leftarrow \theta^t - (1/r) \sum_{i \in \mathcal{W}^t} y^t_i \)
9: end for
10: return \( \theta^T \)

ClientUpdate(\( \theta^t \)):
1: \( \theta^{t,0} \leftarrow \theta^t \)
2: for \( s = 0 \) to \( \tau-1 \) do
3: Compute a mini-batch stochastic gradient \( g^{t,s}_i \)
4: \( \theta^{t,s+1} \leftarrow \theta^{t,s} - \eta_l g^{t,s}_i \)
5: end for
6: \( \Delta^t_i \leftarrow \theta^t - \theta^{t,\tau} \)
7: \( \Delta^t_i \leftarrow \Delta^t_i \times \min(1, C/\|\Delta^t_i\|_2) \)
8: \( \Delta^t_i \leftarrow \Delta^t_i + N(0, (C^2\sigma^2/r) \cdot I_d) \)
9: Encrypt \( \Delta^t_i \) and send it to the server via secure aggregation.

Datasets are defined. Adjacent datasets in the context of DP refer to two datasets that differ by at most one entry. In this paper, we define the adjacent datasets by adding or removing the entire local data of a client and aim to protect the privacy of individual clients in FL, resulting in the client-level DP. Record-level DP, on the other hand, focuses on protecting the privacy of individual records within each client’s local dataset and its adjacent datasets are defined by adding or removing a single datapoint of a client’s local dataset. In comparison, client-level DP can provide a more comprehensive privacy guarantee at the client level and has been shown to be more relevant to the cross-device federated learning we considered, where there are a large number of participating devices, and each device can contribute multiple data records [11].

To provide client-level DP in FL without a fully trusted server, prior studies [9], [11] have proposed the model perturbation mechanism to prevent the privacy leakage from model updates in FedAvg, known as DP-FedAvg. As the pseudo-code of DP-FedAvg given in Algorithm 1, DP-FedAvg has the following changes to FedAvg at each FL round: 1) Each client’s model update is clipped to have a bounded \( \ell_2 \)-norm (line 17); 2) Gaussian noise is added to the clipped model update at each client (line 18); 3) Final local model updates are encrypted following a secure aggregation protocol (e.g., [20], [21]) and sent to the server for aggregation (line 19).

Note that the use of secure aggregation is a common practice in the literature to achieve client-level DP in FL, and the design of a new secure aggregation protocol is out of the scope of this paper. Secure aggregation enables the server to learn just an aggregate function of the clients’ local updates, typically the sum, and nothing else, so the Gaussian noise is added to prevent privacy leakage from the sum of local updates. Specifically, in secure aggregation, clients generate randomly sampled zero-sum mask vectors locally by working in the space of integers modulo \( m \) and sampling the mask’s elements uniformly from \( \mathbb{Z}_m \). When the server computes the modular sum of all the masked updates, the masks cancel out, and the server obtains the exact sum of local updates. As with the existing works in FL [22], [23], [24], we ignore the finite precision and modular summation arithmetic associated with secure aggregation in this paper, noting that one can follow the strategy in [25] to transform the real-valued vectors into integers for minimizing the approximation error of recovering the sum.

Although we can improve privacy and achieve client-level DP in FL by adding Gaussian noise locally and using secure aggregation, the resulting accuracy of the trained model is often low due to the high intensity of added Gaussian noise. Moreover, communication cost has never been considered in those studies. This motivates us to develop a new differentially private FL scheme that can maintain high model accuracy while reducing the communication cost.

IV. Fed-SMP: Federated Learning with Sparsified Model Perturbation

In this section, we develop a new FL scheme called Fed-SMP to provide client-level DP with high model accuracy while improving communication efficiency at the same time. To ensure easy integration into existing packages/systems, Fed-SMP follows the same overall procedure of FedAvg as depicted in Fig. 1 and employs a novel integration of Gaussian mechanism and sparsification in the local update stage. This guarantees that each client’s shared local model update is both sparse and differentially private. The two specific sparsifiers considered in Fed-SMP are defined as follows:

Definition 4 (Random and Top-k Sparsifiers): For a parameter \( 1 \leq k \leq d \) and vector \( x \in \mathbb{R}^d \), the random sparsifier \( \text{rand}_k : \mathbb{R}^d \times \Omega_k \rightarrow \mathbb{R}^d \) and top-k sparsifier \( \text{top}_k : \mathbb{R}^d \rightarrow \mathbb{R}^d \) are defined as

\[
\text{rand}_k(x)_{j} := \begin{cases} x_{j}, & \text{if } j \in \omega, \\ 0, & \text{otherwise}, \end{cases}
\]

\[
\text{top}_k(x) := \begin{cases} x_{\pi(j)}, & \text{if } j \leq k, \\ 0, & \text{otherwise}, \end{cases}
\]

where \( \Omega_k = \binom{[d]}{k} \) denotes the set of all \( k \)-element subsets of \( [d] \), \( \omega \) is chosen uniformly at random, i.e., \( \omega \sim_{\text{unif}} \Omega_k \), and \( \pi \) is a permutation of \( [d] \) such that \( |x_{\pi(j)}| \geq |x_{\pi(j+1)}| \) for \( j \in [1, d - 1] \).

In Fed-SMP with both \( \text{rand}_k \) and \( \text{top}_k \) sparsifier, only \( k \) coordinates of the local model update will be sent out. Generally speaking, \( \text{rand}_k \) sparsifier may discard coordinates that are actually important, which inevitably degrades model accuracy.
when $k$ is small. Yet, $\text{top}_k$ sparsifier keeps the coordinates with the largest magnitude and can achieve higher model accuracy for small $k$. This seems to indicate that $\text{top}_k$ is always the better choice. However, the two sparsifiers have different privacy implications. For $\text{rand}_k$ sparsifier, since the set of selected coordinates $\omega$ is chosen uniformly at random, the selected coordinates are not data-dependent and thus do not contain any private information of training data. On the other hand, the set of selected coordinates for $\text{top}_k$ sparsifier depends on the values of model parameters and hence contains private information of training data and cannot be used like that of $\text{rand}_k$ sparsifier. It is worth noting that although $\text{rand}_k$ sparsifier is a randomized mechanism, it does not provide any privacy guarantee by itself in terms of DP and needs to be combined with Gaussian mechanism for rigorous DP guarantee.

Using the above sparsifiers for client-level DP in FL has new challenges. As mentioned before, secure aggregation is a key privacy-enhancing technique to achieve client-level DP in practical FL systems. However, the naive application of $\text{rand}_k$ or $\text{top}_k$ sparsifier to the local model update of each client typically results in a different set of $k$ coordinates for each client, preventing us from only encrypting the $k$ selected coordinates of each client in the secure aggregation protocol. One can apply secure aggregation to all the $d$ coordinates, but the communication efficiency benefit of sparsification will get lost. In the following, we design new $\text{rand}_k$ and $\text{top}_k$ sparsifiers in Fed-SMP that are compatible with secure aggregation. Specifically, we let the selected clients keep the same set of $k$ active coordinates at each round; therefore, those clients can transmit the sparsified model update directly using the secure aggregation protocol and save the communication cost.

The pseudo-code for the proposed Fed-SMP is provided in Algorithm 2. At the beginning of round $t$, the server randomly selects a set $\mathcal{W}^t$ of $r$ clients and broadcasts to them the current global model $\theta^t$ and mask vector $m^t \in \{0, 1\}^d$ (lines 3-9). The jth coordinate of mask vector $m^t$ equals to 1 if that coordinate is selected by the sparsifier at round $t$ and 0 otherwise. In Fed-SMP with $\text{rand}_k$ sparsifier, the $k$ coordinates are selected randomly from $[d]$ by the server (i.e., the procedure $\text{SelectRandK}(\cdot)$). In Fed-SMP with $\text{top}_k$ sparsifier, we let the server choose a set of top $k$ coordinates for all clients using a small public dataset $D_p$ at each round, avoiding privacy leakage from the selected coordinates. The distribution of the public dataset $D_p$ is assumed to be similar to the overall dataset distribution of clients. This assumption is common in the FL literature [26], [27], [28], [29], where the server is assumed to have a small public dataset for model validation that mimics the overall dataset distribution. Specifically, at round $t$, the server first performs multiple iterations of SGD similar to (3) on the global model $\theta^t$ using the public dataset $D_p$ and obtains the model difference $\Delta_p$. Then, the server selects a set of $k$ coordinates with the largest absolute values in $\Delta_p$ and generates a corresponding mask vector $m^t$ (i.e., the procedure $\text{SelectTopK}(\cdot)$). After receiving the global model $\theta^t$ and mask vector $m^t$ from the server, each client $i \in \mathcal{W}^t$ initializes its local model to $\theta^t$ and runs $\tau$ iterations of SGD to update its local model.
in parallel (lines 26-30). Then, the client $i$ sparsifies its local model update $\theta^{t} - \theta^{t-1}$ using the mask vector $m^{t}$ (lines 33-37). The operator $\odot$ in Algorithm 2 represents the element-wise multiplication. Note that for rand$_k$ sparsifier, the model update will be scaled by $d/k$ to ensure an unbiased estimation on the sparsified model update (line 34). Since there is no a priori bound on the size of the sparsified model update $\Delta^{t}$, each client will clip its sparsified model update in $\ell_2$-norm with clipping threshold $C$ so that $||\Delta^{t}||_2 \leq C$ (line 38). Next, each client perturbs its sparsified model update by adding independent Gaussian noise $\mathcal{N}(0, C^2\sigma^2/r)$ on the $k$ selected coordinates (line 39), where $\sigma$ represents the noise multiplier. The noisy sparsified model update $\Delta^{t}_{i}$ is then encrypted as an input into a secure aggregation protocol and sent to the server (line 32). Finally, the server computes the modular sum of all encrypted models to obtain the exact sum of local model updates (i.e., $\sum_{i\in W} x_i = \sum_{i\in W} \Delta^{t}_{i}$) and updates the global model for the next round (line 13).

V. MAIN THEORETICAL RESULTS

In this section, we analyze the end-to-end privacy guarantee and convergence results of Fed-SMP. For better readability, we state the main theorems and only give the proof sketches in this section while leaving the complete proofs in the Appendix. Before presenting our theoretical results, we make the following assumptions:

**Assumption 1 (Smoothness):** Each local objective function $f_i: \mathbb{R}^d \to \mathbb{R}$ is $L$-smooth, i.e., for any $x, y \in \mathbb{R}^d$, 

$$||\nabla f_i(y) - \nabla f_i(x)|| \leq L||y - x||, \forall i \in [n].$$

**Assumption 2 (Unbiased Gradient and Bounded Variance):** The stochastic gradient at each client is an unbiased estimator of the local gradient: $\mathbb{E}[g_i(x)] = \nabla f_i(x)$, and has bounded variance: $\mathbb{E}[||g_i(x) - \nabla f_i(x)||^2] \leq \tilde{c}_i^2, \forall x \in \mathbb{R}^d, i \in [n]$, where the expectation is over all the local mini-batches. We also denote $\tilde{c}_i^2 := (1/n) \sum_{x=1}^n c_i^2$ for convenience.

**Assumption 3 (Bounded Dissimilarity):** There exist constants $\beta_2 \geq 1, \kappa_2 \geq 0$ such that $(1/n) \sum_{x=1}^n ||\nabla f_i(x)||^2 \leq \beta_2^2 (1/n) \sum_{x=1}^n ||\nabla f_i(x)||^2 + \kappa_2^2$. If local objective functions are identical to each other, then we have $\beta_2^2 = 1$ and $\kappa_2^2 = 0$.

Assumptions 1 and 2 are standard in the analysis of SGD [30], and Assumption 3 is commonly used in the federated optimization literature [31], [32], [33] to capture the dissimilarities of local objectives under non-IID data distribution.

### A. Privacy Analysis

In this subsection, we provide the end-to-end privacy analysis of Fed-SMP based on RDP. Given the fact that the server only knows the sum of local model updates $\sum_{i\in W} \Delta^{t}_{i}$ due to the use of secure aggregation, we need to compute the privacy loss incurred from releasing the sum of local model updates. Assume the client sets $W$ and $W'$ differ in one client index $c$ such that $W' := W \cup \{c\}$. For any adjacent datasets $D := \{D_i\}_{i\in W}$ and $D' := \{D'_j\}_{j\in W'} = \{D_i\}_{i\in W} \cup D_c$, according to Definition 3, the $\ell_2$-sensitivity of the sum of local model updates is

$$\psi := \sup_{D, D'} \left|\sum_{i\in W} \sum_{j\in W} \Delta^{t}_{i}(D_i) - \sum_{j\in W'} \Delta^{t}_{j}(D'_j)\right|_2.$$  

Due to the clipping, we have $||\Delta^{t}_{i}(D_i)||_2 \leq C, \forall i \in [n]$, and therefore $\psi := \sup_{D, D'} ||\sum_{i\in W} \Delta^{t}_{i}(D_i)||_2 \leq C$. As the sum of Gaussian random variables is still a Gaussian random variable, the variance of the Gaussian noise added to each selected coordinate of the sum of local model updates is $C^2\sigma^2$. According to Lemmas 1-4, we compute the overall privacy guarantee of Fed-SMP as follows:

**Theorem 1 (Privacy Guarantee of Fed-SMP):** Suppose the client is sampled without replacement with probability $q := r/n$ at each round. For any $\epsilon < 2\log(1/\delta)$ and $\delta \in (0, 1)$, Fed-SMP satisfies $(\epsilon, \delta)$-DP after $T$ communication rounds if $\sigma^2 \geq \frac{7q^2T(\epsilon + 2\log(1/\delta))}{\epsilon^2}$.

**Proof:** After adding noise $\mathcal{N}(0, C^2\sigma^2)$ to the $k$ selected coordinates, the sum of local model updates satisfies $(\alpha, \alpha/2\sigma^2)$-RDP for each client in $W^t$ at round $t$ by Lemma 2. As $r$ clients are uniformly sampled from all clients at each round, the per-round privacy guarantee of Fed-SMP can be further amplified according to the subsampling amplification property of RDP in Lemma 3. The overall privacy guarantee of Fed-SMP follows by using the composition property in Lemma 4 to compute the RDP guarantee after $T$ rounds and Lemma 1 to convert RDP to $(\epsilon, \delta)$-DP. The details are given in Appendix A, available online. □

### B. Convergence Analysis

In this subsection, we provide the convergence result of Fed-SMP under the general non-convex setting in Theorem 2.

**Theorem 2 (Convergence result of Fed-SMP):** Under Assumptions 1–3, assume the local learning rate satisfies $\eta_t \leq \min((1/24\tau L(\phi_k + 1))^2, 1/4\tau L\sqrt{3/\beta_2^2}, 1/12\tau L)$ and $||\Delta||_2 \leq C$, then the sequence of outputs $\theta^t$ generated by Algorithm 2 satisfies:

$$\frac{1}{T} \sum_{t=0}^{T-1} \left|\nabla f(\theta^t)\right|_2^2 \leq \frac{8e_0}{T\eta_t\tau} + 8\eta_t\tau L(3\kappa_2^2 + 2\tilde{c}_2^2)$$

$$+ (8\eta_t\tau L(2\kappa_2^2 + \tilde{c}_2^2) + \zeta_2^2) \phi_k + \frac{4LkC^2\sigma^2}{\eta_t\tau r^2},$$

where $e_0 := f(\theta^0) - f^*$, $\phi_k := 1 - k/d$ and $\zeta_2 := 4\tilde{c}_2^2/\gamma$ for top$_k$ sparsifier, $\phi_k := d/k - 1$ and $\zeta_2 := 0$ for rand$_k$ sparsifier, and $f^*$ represents the optimal objective value.

**Proof:** The proof is given in Appendix C and D, available online. □

The convergence bound (7) contains three parts. The first two terms $8e_0/T\eta_t\tau + 8\eta_t\tau L(3\kappa_2^2 + 2\tilde{c}_2^2)$ represent the optimization error bound in FedAvg. The third term $(8\eta_t\tau L(2\kappa_2^2 + \tilde{c}_2^2) + \zeta_2^2)\phi_k$ is the compression error resulted from applying sparsification on local model updates. The last term $4LkC^2\sigma^2/\eta_t\tau r^2$ is the privacy error resulted from adding DP noise to perturb
local model updates. Both compression error and privacy error increase the error floor at convergence. When no sparsification is applied (i.e., $k = d$ and $\phi_k = 0$), the compression error is zero. When no DP noise is added (i.e., $\sigma = 0$), the privacy error is zero. The above result shows an explicit tradeoff between compression error and privacy error in Fed-SMP. As $k$ decreases, the variance of sparsification $\phi_k$ gets larger which leads to a larger compression error, but the privacy error decreases. Therefore, there exists an optimal parameter $k$ that can balance those two errors to minimize the convergence bound.

VI. PERFORMANCE EVALUATION

In this section, we evaluate the performance of Fed-SMP with rand$k$ and top$k$ sparsifiers (denoted by Fed-SMP-rand$k$ and Fed-SMP-top$k$, respectively) by comparing it with the following baselines:
- **FedAvg**: The classic FL algorithm served as the baseline without privacy consideration;
- **FedAvg-top$k$**: A communication-efficient variant of FedAvg, where the local model update is compressed using top$k$ sparsifier before being uploaded;
- **FedAvg-rand$k$**: Another communication-efficient variant of FedAvg, where the local model update of each client is compressed using rand$k$ sparsifier before being uploaded;
- **DP-FedAvg [11]**: The state-of-art differentially-private variant of FedAvg that achieves client-level DP, where the full-precision local model update from each client is clipped with clipping threshold $C$ and then perturbed by adding Gaussian noise drawn from the distribution $\mathcal{N}(0, (C^2\sigma^2/r) \cdot \mathbf{I}_d)$, where $\sigma$ is the noise multiplier and $r$ is the number of selected clients per round.

For fair comparisons, all the algorithms use the same secure aggregation protocol. When $p = 1.0$, Fed-SMP-top$k$ and Fed-SMP-rand$k$ reduce to DP-FedAvg, and FedAvg-top$k$ and FedAvg-rand$k$ reduce to FedAvg.

A. Dataset and Model

We use Fashion-MNIST [34] dataset, SVHN dataset [35] and Shakespeare dataset [5], three common benchmarks for differentially private machine/federated learning. The first two are image classification datasets, and the last one is a text dataset for next-character-prediction. Note that while Fashion-MNIST and SVHN datasets are considered as “solved” in the computer vision community, achieving high accuracy with strong privacy guarantee remains difficult on these datasets [36].

The Fashion-MNIST dataset consists of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a $28 \times 28$ grayscale image associated with a label from 10 classes. For Fed-SMP-top$k$, we randomly sample 1,000 examples from the training set as the public dataset $D_p$ and evenly partition the remaining 59,000 examples across 6,000 clients. For other algorithms, we evenly split the training data across 6,000 clients. We use the convolutional neural network (CNN) model in [5] on Fashion-MNIST that consists of two $5 \times 5$ convolution layers (the first with 32 filters, the second with 64 filters, each followed by $2 \times 2$ max pooling and ReLu activation), a fully connected layer with 512 units and ReLu activation, and a final softmax output later, which results in about 1.6 million total parameters.

The SVHN dataset contains 73,257 training examples and 26,032 testing examples, and each of them is a $32 \times 32$ colored image of digits from 0 to 9. We extend the training set with another 531,131 additional examples. For all algorithms except Fed-SMP-top$k$, we evenly partition the training set across 6,000 clients. For Fed-SMP-top$k$, we randomly sample 2,000 examples from the training set as the public dataset $D_p$ and evenly partition the remaining across 6,000 clients. We train a CNN model used in [37] on SVHN dataset, which stacks two $5 \times 5$ convolution layers (the first with 64 filters, the second with 128 filters, each followed with $2 \times 2$ max pooling and ReLu activation), two fully connected layers (the first with 384 units, the second with 192 units, each followed with ReLu activation), and a final softmax output later, resulting in about 3.4 million parameters in total.

The Shakespeare dataset is a natural federated dataset built from The Complete Works of William Shakespeare, where each of the total 715 clients corresponds to a speaking role with at least two lines. The training set and test set are obtained by splitting the lines from each client, and each client has at least one line for training or testing. We refer the reader to [38] for more details on processing the raw data. Note that for Fed-SMP-top$k$, we randomly select 1000 samples in total from the processed training set of clients as the public dataset $D_p$. For Shakespeare, we use the recurrent neural network (RNN) model in [38] to predict the next work based on the preceding words in a line. The RNN model takes an 80-character sequence as input and consists of a $80 \times 8$ embedding layers and 2 LSTM layers (each with 256 nodes) followed by a dense layer, resulting in about 0.8 million parameters in total.

B. Experimental Settings

For Fashion-MNIST and SVHN datasets, the server randomly samples $r = 100$ clients to participate in the training at each round, and for Shakespeare, $r = 10$. For the local optimizer on clients, we use momentum SGD for both datasets. Specifically, for Fashion-MNIST, we set the local momentum coefficient as 0.5, local learning rate as $\eta_l = 0.125$ decaying at a rate of 0.99 at each communication round, batch size $B = 10$ and local update period to be 10 epochs; for SVHN, we set the local momentum coefficient as 0.8, local learning rate as $\eta_l = 0.05$ decaying at a rate of 0.99 at each communication round, batch size $B = 50$, local update period to be 5 epochs and number of communication rounds $T = 180$; for Shakespeare, we set the local momentum coefficient as 0.9, local learning rate as $\eta_l = 1$ decaying at a rate of 0.99 at every 50 round, batch size $B = 4$ and local update period to be 1 epoch and number of rounds $T = 1000$. It is worth noting that the momentum at each client is initialized at the beginning of every communication round, since local momentum will be stale due to the partial participation of clients. Specially, for Fed-SMP-top$k$, the server uses the same optimizer as the clients with the same local update period to compute the global model update $\Delta_p$. 

For the privacy-preserving algorithms, we compute the end-to-end privacy loss using the API provided in [39]. For all datasets, the privacy parameter \( \delta \) is set to be \( \delta < 1/n \).

For Fashion-MNIST and SVHN datasets, we follow [11] to set \( \delta = 1/n \).1 where \( n = 6000 \) is the number of clients; for Shakespeare dataset, we follow [40] and [11] to set \( \delta = 1/n \).1.1, where \( n = 10^9 \) is a hypothetical population size for ease of reporting privacy results. Moreover, the noise multiplier is set as \( \sigma = 1.4 \) for Fashion-MNIST and SVHN datasets and \( \sigma = 0.3 \) for Shakespeare dataset by default. We tune the clipping threshold over the grid \( C = \{0.1, 0.2, 0.4, 0.6, 0.8, 1.0, 2.0\} \) and obtain the optimal clipping threshold, i.e., \( C = 1.0 \) for Fashion-MNIST dataset, \( C = 0.4 \) for SVHN and Shakespeare datasets.

In addition, we define the compression ratio for \( \text{rand}_k \) and \( \text{top}_k \) sparsifiers as \( p := k/d \). When \( p \) is smaller, more coordinates of the local model updates are set to zero, leading to a higher level of communication compression.

### C. Experiment Results

Tables I, II, and III summarize the performance of FedAvg, DP-FedAvg and Fed-SMP after \( T \) rounds on Fashion-MNIST, SVHN and Shakespeare, respectively. We run each experiment with 5 random seeds and report the average and standard deviation of testing accuracy. Note that we always report the best testing accuracy across all rounds in each experiment. We also report the uplink communication cost that denotes the size of the data sent from a client to the server as the input into the secure aggregation protocol, which equals to \( 32 k \times T \times (r/n) \) bits for Fed-SMP and \( 32 \times T \times (r/n) \) bits for DP-FedAvg and FedAvg.

#### Table I: Summary of Results on Fashion-MNIST Dataset

| Compression ratio | Algorithm     | Accuracy (%) | Cost (MB) | Privacy (\( \epsilon \)) |
|-------------------|---------------|--------------|-----------|---------------------------|
| \( p = 0.001 \)   | Fed-SMP-top_\( k \) | 77.18 ± 0.48 | 0.02      | 1.00                      |
|                   | Fed-SMP-rand_\( k \) | 27.16 ± 10.20 | 0.02      | 1.00                      |
| \( p = 0.005 \)   | Fed-SMP-top_\( k \) | 80.76 ± 0.27 | 0.10      | 1.00                      |
|                   | Fed-SMP-rand_\( k \) | 56.64 ± 5.22 | 0.10      | 1.00                      |
| \( p = 0.01 \)    | Fed-SMP-top_\( k \) | 80.76 ± 0.47 | 0.20      | 1.00                      |
|                   | Fed-SMP-rand_\( k \) | 65.61 ± 1.04 | 0.20      | 1.00                      |
| \( p = 0.1 \)     | Fed-SMP-top_\( k \) | 80.49 ± 0.23 | 2.00      | 1.00                      |
|                   | Fed-SMP-rand_\( k \) | 77.59 ± 0.53 | 2.00      | 1.00                      |
| \( p = 0.2 \)     | Fed-SMP-top_\( k \) | 80.16 ± 0.09 | 3.99      | 1.00                      |
|                   | Fed-SMP-rand_\( k \) | 79.12 ± 0.40 | 3.99      | 1.00                      |
| \( p = 0.4 \)     | Fed-SMP-top_\( k \) | 80.26 ± 0.25 | 7.98      | 1.00                      |
|                   | Fed-SMP-rand_\( k \) | 79.88 ± 0.37 | 7.98      | 1.00                      |
| \( p = 0.8 \)     | Fed-SMP-top_\( k \) | 79.77 ± 1.00 | 15.97     | 1.00                      |
|                   | Fed-SMP-rand_\( k \) | 79.82 ± 0.35 | 15.97     | 1.00                      |
| \( p = 1.0 \)     | DP-FedAvg     | 72.72 ± 3.72 | 19.96     | 1.00                      |
|                   | FedAvg        | 86.98 ± 0.12 | 19.96     | -                         |

#### Table II: Summary of Results on SVHN Dataset

| Compression ratio | Algorithm     | Accuracy (%) | Cost (MB) | Privacy (\( \epsilon \)) |
|-------------------|---------------|--------------|-----------|---------------------------|
| \( p = 0.005 \)   | Fed-SMP-top_\( k \) | 80.94 ± 0.85 | 0.21      | 1.00                      |
|                   | Fed-SMP-rand_\( k \) | 19.45 ± 0.42 | 0.21      | 1.00                      |
| \( p = 0.01 \)    | Fed-SMP-top_\( k \) | 81.12 ± 1.08 | 0.41      | 1.00                      |
|                   | Fed-SMP-rand_\( k \) | 19.75 ± 0.12 | 0.41      | 1.00                      |
| \( p = 0.05 \)    | Fed-SMP-top_\( k \) | 80.22 ± 1.02 | 2.06      | 1.00                      |
|                   | Fed-SMP-rand_\( k \) | 28.15 ± 2.46 | 2.06      | 1.00                      |
| \( p = 0.1 \)     | Fed-SMP-top_\( k \) | 79.64 ± 0.29 | 4.12      | 1.00                      |
|                   | Fed-SMP-rand_\( k \) | 64.50 ± 2.95 | 4.12      | 1.00                      |
| \( p = 0.2 \)     | Fed-SMP-top_\( k \) | 77.91 ± 0.37 | 8.24      | 1.00                      |
|                   | Fed-SMP-rand_\( k \) | 76.94 ± 0.58 | 8.24      | 1.00                      |
| \( p = 0.4 \)     | Fed-SMP-top_\( k \) | 75.58 ± 0.40 | 16.47     | 1.00                      |
|                   | Fed-SMP-rand_\( k \) | 77.07 ± 0.48 | 16.47     | 1.00                      |
| \( p = 0.8 \)     | Fed-SMP-top_\( k \) | 73.18 ± 0.69 | 32.94     | 1.00                      |
|                   | Fed-SMP-rand_\( k \) | 73.90 ± 0.37 | 32.94     | 1.00                      |
| \( p = 1.0 \)     | DP-FedAvg     | 72.46 ± 0.54 | 41.18     | 1.00                      |
|                   | FedAvg        | 88.47 ± 0.17 | 41.18     | -                         |

#### Table III: Summary of Results on Shakespeare Dataset

| Compression ratio | Algorithm     | Accuracy (%) | Cost (MB) | Privacy (\( \epsilon \)) |
|-------------------|---------------|--------------|-----------|---------------------------|
| \( p = 0.005 \)   | Fed-SMP-top_\( k \) | 55.41 ± 0.20 | 0.23      | 6.99                      |
|                   | Fed-SMP-rand_\( k \) | 36.41 ± 0.15 | 0.23      | 6.99                      |
| \( p = 0.01 \)    | Fed-SMP-top_\( k \) | 56.04 ± 0.23 | 0.46      | 6.99                      |
|                   | Fed-SMP-rand_\( k \) | 38.99 ± 0.73 | 0.46      | 6.99                      |
| \( p = 0.05 \)    | Fed-SMP-top_\( k \) | 56.76 ± 0.13 | 2.30      | 6.99                      |
|                   | Fed-SMP-rand_\( k \) | 50.79 ± 0.27 | 2.30      | 6.99                      |
| \( p = 0.1 \)     | Fed-SMP-top_\( k \) | 55.74 ± 0.06 | 4.60      | 6.99                      |
|                   | Fed-SMP-rand_\( k \) | 53.15 ± 0.14 | 4.60      | 6.99                      |
| \( p = 0.2 \)     | Fed-SMP-top_\( k \) | 55.89 ± 0.14 | 9.20      | 6.99                      |
|                   | Fed-SMP-rand_\( k \) | 54.22 ± 0.15 | 9.20      | 6.99                      |
| \( p = 0.4 \)     | Fed-SMP-top_\( k \) | 54.36 ± 0.22 | 18.41     | 6.99                      |
|                   | Fed-SMP-rand_\( k \) | 53.68 ± 0.14 | 18.41     | 6.99                      |
| \( p = 0.8 \)     | Fed-SMP-top_\( k \) | 52.14 ± 0.31 | 36.82     | 6.99                      |
|                   | Fed-SMP-rand_\( k \) | 51.85 ± 0.29 | 36.82     | 6.99                      |
| \( p = 1.0 \)     | DP-FedAvg     | 51.03 ± 0.28 | 46.02     | 6.99                      |
|                   | FedAvg        | 62.77 ± 0.05 | 46.02     | -                         |
Fig. 2. Testing accuracy (%) of Fed-SMP and DP-FedAvg w.r.t. communication round (left) and uplink communication cost (right).

and Fed-SMP-randk achieves a testing accuracy of 54.22% on Shakespeare when \( p = 0.2 \), outperforming DP-FedAvg by 6%. In the following, we further evaluate Fed-SMP with more experimental settings.

1) Communication Efficiency of Fed-SMP: For the communication cost, we can observe from Tables I, II, and III that the uplink communication cost for Fed-SMP on both datasets is relatively lower than that of DP-FedAvg and FedAvg under the same privacy guarantee, since our scheme integrates well with the secure aggregation protocol and takes advantage of model update compression. To further demonstrate the communication efficiency of Fed-SMP, we show the convergence speed and communication cost of our Fed-SMP algorithms. Specifically, we select the Fed-SMP algorithms with the optimal compression ratio and show its testing accuracy with respect to the communication round and uplink communication cost in Fig. 2, compared with DP-FedAvg and FedAvg; For Fashion-MNIST dataset, we draw the results of Fed-SMP-topk with \( p = 0.005 \) and Fed-SMP-randk with \( p = 0.4 \), while achieving \((1.01, n^{-1.1})\)-DP; for SVHN dataset, we draw the results of Fed-SMP-topk with \( p = 0.01 \) and Fed-SMP-randk with \( p = 0.4 \), while achieving \((1.01, n^{-1.1})\)-DP; for Shakespeare dataset, we draw the results of Fed-SMP-topk with \( p = 0.05 \) and Fed-SMP-randk with \( p = 0.2 \), while achieving \((6.99, n^{-1.1})\)-DP.

From Fig. 2, we observe that DP-FedAvg converges to a lower accuracy than FedAvg due to the added DP noise. Fed-SMP converges slower than DP-FedAvg due to the use of sparsification. However, the final accuracy of Fed-SMP is higher than DP-FedAvg because of the advantage of sparsification in improving privacy and model accuracy. Moreover, Fig. 2 also demonstrates that Fed-SMP is more communication-efficient than DP-FedAvg, and Fed-SMP-topk is more communication-efficient than Fed-SMP-randk. For instance, to achieve a target accuracy 72% on Fashion-MNIST dataset, Fed-SMP-topk and Fed-SMP-randk save 99% and 70% of uplink communication cost compared with DP-FedAvg, respectively; to achieve a target accuracy 72% on SVHN dataset, Fed-SMP-topk and Fed-SMP-randk save 99% and 82% of uplink communication cost compared with DP-FedAvg, respectively; to achieve a target accuracy 50% on Shakespeare dataset, Fed-SMP-topk and
Fed-SMP-rand_k save 95% and 60% of uplink communication cost compared with DP-FedAvg, respectively.

2) Tradeoff Between Privacy and Compression Errors: From Tables I, II, and III, we have observed that as compression ratio \( p \) decreases from 1.0 to a small value (e.g., 0.005), the testing accuracy of Fed-SMP first increases and then decreases. This is due to the change of privacy error and compression error in the convergence bound (7), which is controlled by \( k \). Since the compression ratio \( p = k/d \), the smaller \( p \) is, the lower the privacy error is, but the compression error could increase. In the following, we conduct additional experiments to further demonstrate this tradeoff under various compression ratios. As it is hard to directly show the tradeoff between privacy error and compression error, we study the difference between the performances of Fed-SMP and FedAvg-rand_k/top_k, which differ only in DP noise addition.

Specifically, we show the training losses and testing accuracies of Fed-SMP, FedAvg-rand_k, and FedAvg-top_k with respect to different compression ratios in Fig. 3. Note that here Fed-SMP achieves (1.01, \( n^{-1.1} \))-DP for Fashion-MNIST and SVHN datasets and (6.99, \( n^{-1.1} \))-DP for Shakespeare dataset. From the results, we can see that as the compression ratio \( p \) increases, the training losses of FedAvg-rand_k and FedAvg-top_k on all datasets monotonically decrease, and the corresponding testing accuracies monotonically increase, due to the decreasing compression error incurred from sparsification. However, the training losses of Fed-SMP-rand_k and Fed-SMP-top_k decrease at first and then increase, and the corresponding testing accuracies increase at first and then decrease. This matches our observations from Theorem 2, i.e., the performance of Fed-SMP depends on the tradeoff between the compression error and privacy error. When \( p \) is too small (e.g., \( p < 0.1 \) for Fed-SMP-rand_k on Fashion-MNIST datasets), despite the privacy amplification effect of compression that reduces the amount of added Gaussian noise, the compression error is significant and dominates the total convergence error, leading to a higher training loss and lower testing accuracy. As \( p \) increases, the compression error decreases, leading to a lower training loss. However, if \( p \) becomes too large, the privacy amplification effect of compression becomes negligible, and the privacy error starts to dominate the convergence error, leading to a higher training loss. For example, the testing accuracies of Fed-SMP-rand_k start to decrease when \( p > 0.4 \) for Fashion-MNIST and SVHN datasets and \( p > 0.2 \) for Shakespeare dataset, and the testing accuracy of Fed-SMP-top_k starts to decrease when \( p > 0.005 \) on Fashion-MNIST dataset, \( p > 0.01 \) on SVHN dataset, and
p > 0.05 for Shakespeare dataset. Therefore, to achieve the best performance of Fed-SMP, the choice of p needs to balance the privacy error and compression error. Furthermore, we find that Fed-SMP-topk tends to achieve its best performance when p is small (i.e., 0.005, 0.01 and 0.1 for Fashion-MNIST, SVHN and Shakespeare datasets, respectively), and Fed-SMP-randk tends to achieve its best performance when p is large (i.e., 0.4, 0.4, 0.2 for Fashion-MNIST, SVHN and Shakespeare datasets, respectively).

3) Tradeoff Between Privacy and Accuracy: In this part, we study the tradeoff between privacy and model accuracy of Fed-SMP. In Tables IV–VII, we show the testing accuracy of Fed-SMP with respect to the privacy loss, under different compression ratios. Note that here we vary the noise multiplier σ to achieve different privacy losses. We can see that under the same compression ratio p, the testing accuracy of Fed-SMP always decreases when the privacy loss ϵ decreases. For example, the testing accuracy of Fed-SMP-randk with p = 0.005 decreases from 60.02% to 51.72% on Fashion-MNIST dataset when the privacy loss ϵ decreases from 2.02 to 0.44. The reason is that as ϵ decreases, the noise multiplier σ increases (according to Theorem 1), and hence, the convergence error increases (according to Theorem 2). Furthermore, we also observe that under the same privacy loss, the testing accuracy of Fed-SMP always increases at first and then decreases, as the compression ratio p increases. For example, when the privacy loss ϵ = 2.02, the testing accuracy of Fed-SMP-randk on SVHN dataset increases from 28.18% to 81.35% and then decreases to 80.39%, as the compression ratio p increases from 0.05 to 0.8. This matches with our observations from Fig. 3, i.e., there exists an optimal compression ratio that achieves the highest testing accuracy under the same privacy loss (as highlighted in the table) due to the tradeoff between privacy and compression errors.

Besides, the optimal compression ratio decreases as the privacy loss decreases. For example, from Table IX, the optimal p for Fed-SMP-topk on Shakespeare dataset is 0.1 when ϵ = 11.51, and as ϵ decreases to 3.39, the optimal p decreases to 0.01. We observe similar trends on Tables IV–VIII. It is because that the optimal p always balances the tradeoff between privacy and compression errors to minimize the overall convergence error. As ϵ decreases, the noise multiplier σ increases, and thus, the privacy error increases. In this case, the optimal p should decrease to reduce the privacy error.

VII. RELATED WORK

FL, or distributed learning in general, with DP has attracted increasing attention recently. Different from centralized learning, FL involves multiple communication rounds between clients and the server; therefore, DP needs to be preserved for all communication rounds. The main challenge of achieving DP in FL lies in maintaining high model accuracy under a reasonable DP guarantee.

In FL, DP can be categorized into record-level DP [41], [42] and client-level DP [9], [11], depending on the granularity.
of privacy protection. Client-level DP provides more realistic protections than record-level DP against information leakage in the FL setting. With client-level DP, the entire local dataset of a client rather than a single data record needs to be protected, therefore requiring the addition of a larger amount of noise than record-level DP to achieve the same level of DP [11], [43]. Our work follows this line of research and proposes to integrate SMP with FedAvg to achieve higher model accuracy than prior schemes under the same client-level DP guarantee in FL. Note there are other approaches to preserve privacy in distributed learning, such as secure multi-party computation [44] or homomorphic encryption [45]. However, those approaches have different privacy goals and cannot protect against information leakage incurred from observing the final trained models at the server.

Prior related works that design DP mechanisms together with some specialized techniques such as shuffling [12], [13], [46], quantization [47], [48], [49], [50], sparsification [14], [15], [16], low-rank approximation [51], [52], [53], compressive sensing [54], and aggregation directly over wireless channels [55], either are orthogonal to our works and only applicable to specific settings or cannot boost privacy and improve communication efficiency simultaneously. For instance, the model shuffling method in [12] uses a server to permute the local model updates in FL before aggregation, improving the DP guarantee for protecting the local model update of each client, which is orthogonal to our work that aims to protect the local dataset of each client and could be combined to achieve better privacy protections. The compressive sensing method in [54] reduces the model size before adding DP noise to model updates in FL but relies on the strong assumption that model updates only have a few non-zero coordinates, which seldom holds in practice.

The quantization-based methods [47], [48], [49], [50] aim to design a feasible DP mechanism that is compatible with quantization to achieve both DP guarantee and communication efficiency. For example, [47] aims to design a binomial DP mechanism for stochastic gradient quantization so that the perturbed gradients still have low precision after adding DP noises, and [48] and [49] extend this work using the discrete Gaussian mechanism. Although these works achieve DP and improve communication efficiency simultaneously, the use of quantization cannot reduce the impact of DP noise. However, in our method, the use of sparsification can also reduce the impact of DP noise so that we can improve the model utility and achieve high communication efficiency and DP simultaneously.

The sparsification-based methods reduce the impact of DP noises by reducing the model dimension using sparsification [14], [15], [16], [56]. For example, [56] uses top $k$ sparsification and stochastic sign quantization to compress the gradient before adding DP noise. However, this method is designed to protect each gradient vector in a centralized setting, while our method is designed to provide comprehensive and direct DP protection to the local dataset in a distributed setting. In [14], the top $k$ sparsification is integrated with record-level DP to protect the local model update of a client, and in [16] random sparsification is used with gradient perturbation to achieve record-level DP. Our approach is orthogonal to theirs as we aim to achieve client-level DP that protects the whole local dataset of a client comprehensively. Moreover, our work theoretically analyzes the convergence of the proposed methods to demonstrate the benefit of integrating biased or unbiased sparsification with DP in the classic FedAvg algorithm under non-IID data distribution, filling the gap in the state-of-arts. Note that Zhang et al. [57] also combine sparsification with Gaussian mechanism to achieve communication efficiency and DP in decentralized SGD, but their scheme adds noise before using sparsification and cannot leverage sparsification to improve privacy.

The low-rank approximation-based method [51], [52], [53] is to map the model to a lower-dimensional subspace so that fewer DP noises are needed, and this is orthogonal to our model sparsification-based method. For example, in both [51] and [52], the noisy gradient is projected to a low-dimensional subspace to reduce the size of DP noise. However, these methods are designed for centralized ML settings. In [53], the low-rank matrix approximation is used in FL settings to reduce the dimension of local model updates before adding DP noise, but it requires two rounds of communications for each model aggregation. Compared with [53], our method only makes lightweight modifications to FedAvg, the most common learning method for FL, which only requires one round of low-cost communication for each aggregation and enables higher communication efficiency and easier integration into existing packages.

Some other works [58], [59] also develop differentially private versions of distributed algorithms that require fewer iterations than SGD-based algorithms to converge such as alternating direction method of multipliers, but they are only applicable to convex settings and do not allow partial client participation at each communication round, which is a key feature in FL.

Besides privacy, another bottleneck in FL is the high communication cost of transmitting model parameters. Typical techniques to address this issue include quantization and sparsification [60], [61], [62], [63], [64]. For example, in [63], sparsification and quantization are integrated with secure aggregation to mitigate the communication cost and privacy concern of FL simultaneously; in [64] the top $k$ coordinates of a client’s gradient are selected cyclically for all clients to do gradient sparsification to save the communication cost in distributed learning. However, these works do not consider rigorous privacy protection for clients.

**VIII. CONCLUSION**

This paper has proposed Fed-SMP, a new FL scheme based on sparsified model perturbation. Fed-SMP achieves client-level DP while maintaining high model accuracy and is also communication-efficient. We have rigorously analyzed the convergence and end-to-end DP guarantee of the proposed scheme and extensively evaluated its performance on three common benchmark datasets. Experimental results have shown that Fed-SMP with both rand$_k$ and top$_k$ sparsification strategies can improve the privacy-accuracy tradeoff and communication efficiency simultaneously compared with the existing methods. In the future, we will consider other compression methods such as low-rank approximation and quantization, and extend the scheme to alternative FL settings such as shuffled model.
REFERENCES

[1] A. Pothitis, “IoT and wearables: Fitness tracking.” 2017. [Online]. Available: http://www.mobileindustryreview.com/2017/03/iot-wearables-fitness-tracking.html

[2] P. Goldstein, “Smart city gains efficiencies from IoT traffic sensors and data.” 2018. [Online]. Available: https://statetechmagazine.com/article/2018/12/smart-cities-gain-efficiencies-iot-traffic-sensors-and-data-perform

[3] A. Weirenc, “The future of the smart home: Smart homes and IoT: A century in the making.” 2018. [Online]. Available: https://statetechmagazine.com/article/2018/12/smart-cities-gain-efficiencies-iot-traffic-sensors-and-data-perform

[4] P. Kairouz et al., “Advances and open problems in federated learning.” Found. Trends Mach. Learn., vol. 14, no. 1–2, pp. 1–210, 2021.

[5] R. Sharma et al., “Federated learning with noisy user feedback.” 2022, arXiv:2205.03092.

[6] L. Bottou, F. E. Curtis, and J. Nocedal, “Optimization methods for large-scale machine learning.” SIAM Rev., vol. 60, no. 2, pp. 223–311, 2018.

[7] R. Ward, X. Wu, and L. Bottou, “AdaGrad stepizes: Sharp convergence over nonconvex landscapes,” in Proc. Int. Conf. Mach. Learn., 2019, pp. 6677–6686.

[8] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, “On the convergence of FedAvg on Non-IID data,” in Proc. Int. Conf. Learn. Representations, 2020.

[9] Y. Guo, Y. Sun, R. Hu, and Y. Gong, “Hybrid local SGD for federated learning with heterogeneous communications,” in Proc. Int. Conf. Learn. Representations, 2022.

[10] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms.” 2017, arXiv:1708.07747.

[11] R. C. Geyer, T. Klein, and M. Nabi, “Differentially private federated learning in the shuffle model,” in Proc. Adv. Neural Inf. Process. Syst., 2019, pp. 3051–3061.

[12] R. Liu, Y. Cao, M. Yoshikawa, and H. Chen, “FedSel: Federated SGD under local differential privacy with top-k dimension selection,” in Proc. Int. Conf. Database Syst. Adv. Appl., 2020, pp. 485–501.

[13] L. Lyu, “DP-SIGNSGD: When efficiency meets privacy and robustness,” in Proc. IEEE Int. Conf. Acoust. Speech Signal Process., 2021, pp. 3070–3074.

[14] R. Hu, Y. Gong, and Y. Guo, “Federated learning with sparsification-amplified privacy and adaptive optimization,” in Proc. IEEE Int. Conf. Learn. Representations, 2021.

[15] I. Mironov, “Rényi differential privacy,” in Proc. IEEE 30th Comput. Secur. Found. Symp., 2017, pp. 263–275.

[16] Y. Wang, B. Balle, and S. P. Kasiviswanathan, “Subsampled Rényi differential privacy and analytical moments accountant,” in Proc. 22nd Int. Conf. Artif. Intell. Statist., 2019, pp. 1226–1235.

[17] L. Wang, B. Jayaraman, D. Evans, and Q. Gu, “Efficient privacy-preserving nonconvex optimization,” 2020, arXiv:1910.13659.

[18] K. Bonawitz et al., “Practical secure aggregation for privacy-preserving machine learning,” in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., 2017, pp. 1175–1191.

[19] K. Bonawitz et al., “Towards federated learning at scale: System design,” Proc. Mach. Learn. Syst., vol. 1, pp. 374–388, 2019.

[20] S. Goryczka, L. Xiong, and V. Sunderam, “Secure multiparty aggregation with differential privacy: A comparative study,” in Proc. Joint EDBT/ICDT Workshops, 2013, pp. 155–163.

[21] S. Truex et al., “A hybrid approach to privacy-preserving federated learning,” in Proc. 12th ACM Workshop Artif. Intell. Secur., 2019, pp. 1–11.

[22] F. Valovich and F. Alda, “Computational differential privacy from lattice-based cryptography,” in Proc. Int. Conf. Number-Theoretic Methods Cryptol., 2017, pp. 121–141.

[23] K. Bonawitz, F. Salehi, J. Konečný, B. McMahan, and M. Gruteser, “Federated learning with autotuned communication-efficient secure aggregation,” in Proc. ACM SIGSAC Conf. Security Syst. Commun., 2019, pp. 1222–1226.

[24] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, and U. Erlingsson, “Scalable private learning with PATE,” in Proc. Int. Conf. Learn. Representations, 2018.
[54] R. Kerkouche, G. Ács, C. Castelluccia, and P. Genevès, “Compression boosts differentially private federated learning,” in Proc. IEEE 6th Eur. Symp. Secur. Privacy, 2021, pp. 304–318.

[55] M. Seif, R. Tandon, and M. Li, “Wireless federated learning with local differential privacy,” in Proc. IEEE Int. Symp. Inf. Theory, 2020, pp. 2604–2609.

[56] B. Wang, F. Wu, Y. Long, L. Rimanic, C. Zhang, and B. Li, “DataLens: Scalable privacy preserving training via gradient compression and aggregation,” in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., 2021, pp. 2146–2168.

[57] X. Zhang, M. Fang, J. Liu, and Z. Zhu, “Private and communication-efficient edge learning: A sparse differential Gaussian-masking distributed SGD approach,” in Proc. 21st Int. Symp. Theory Algorithmic Found. Protocol Des. Mobile Netw. Mobile Comput., 2020, pp. 261–270.

[58] Y. Guo and Y. Gong, “Practical collaborative learning for crowdsensing in the Internet of Things with differential privacy,” in Proc. IEEE Conf. Commun. Netw. Secur., 2018, pp. 1–9.

[59] Z. Huang, R. Hu, Y. Guo, E. Chan-Tin, and Y. Gong, “DP-ADMM: ADMM-based distributed learning with differential privacy,” IEEE Trans. Inf. Forensics Secur., vol. 15, pp. 1002–1012, 2019.

[60] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, “QSGD: Communication-efficient SGD via gradient quantization and encoding,” in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 1709–1720.

[61] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar, “signSGD: Compressed optimisation for non-convex problems,” in Proc. Int. Conf. Mach. Learn., 2018, pp. 560–569.

[62] H. Wang, S. Sievert, S. Liu, Z. Charles, D. Papailiopoulos, and S. Wright, “ATOMO: Communication-efficient learning via atomic sparsification,” in Proc. Adv. Neural Inf. Process. Syst., 2018, pp. 9850–9861.

[63] C. Beguier, M. Androux, and E. W. Tramel, “Efficient sparse secure aggregation for federated learning,” 2020, arXiv: 2007.14861.

[64] C.-Y. Chen et al., “ScaleCom: Scalable sparsified gradient compression for communication-efficient distributed training,” in Proc. Adv. Neural Inf. Process. Syst., 2020, pp. 13551–13563.

[65] M. Danilova et al., “Recent theoretical advances in non-convex optimization,” in High-Dimensional Optimization and Probability: With a View Towards Data Science. Berlin, Germany: Springer, 2022, pp. 79–163.

[66] J. H. Bell, K. A. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova, “Secure single-server aggregation with (poly) logarithmic overhead,” in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., 2020, pp. 1253–1269.

[67] W. Diffie and M. E. Hellman, “New directions in cryptography,” in Proc. Democratizing Cryptogr.: Work Whitfield Diffie Martin Hellman, 2022, pp. 365–390.

[68] A. Shamir, “How to share a secret,” Commun. ACM, vol. 22, no. 11, pp. 612–613, 1979.

Rui Hu (Member, IEEE) received the BEng degree in electrical engineering from Jinan University, Guangdong, China, in 2017, and the PhD degree in electrical and computer engineering from the University of Texas at San Antonio, San Antonio, TX, USA, in 2022. She is currently an Assistant Professor in the Department of Computer Science and Engineering with the University of Nevada, Reno, Reno, NV, USA. Her research interests include cyber-security and data privacy, federated learning, edge computing, and Internet-of-Things. She received the Dorrough Distinguished Graduate Fellowship and Graduate Student Professional Development Awards in 2018 and 2019, respectively, and the Outstanding Graduate Research Award and Outstanding PhD Dissertation Award of UTSA, in 2022.

Yuaxiong Guo (Senior Member, IEEE) received the BEng degree in electronics and information engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2009, and the MS and PhD degrees in electrical and computer engineering from the University of Florida, Gainesville, FL, USA, in 2012 and 2014, respectively. He is currently an Associate Professor in the Department of Information Systems and Cyber Security, Department of Electrical and Computer Engineering, and School of Data Science with the University of Texas at San Antonio, San Antonio, TX, USA. His research focuses on the intersection of machine learning and networked systems with the goal of enabling pervasive and trustworthy edge intelligence. He is a recipient of the Internal Research Award at UTSA and Best Paper Award with the IEEE Global Communications Conference. He is on the editorial board of IEEE Transactions on Vehicular Technology and has served as the track co-chair for IEEE VTC 2021-Fall, and PST 2022.

Yannmin Gong (Senior Member, IEEE) received the BEng degree in electronics and information engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2009, the MS degree in electrical engineering from Tsinghua University, Beijing, China, in 2012, and the PhD degree in electrical and computer engineering from the University of Florida, Gainesville, FL, USA, in 2016. She is currently an Associate Professor in the Department of Electrical and Computer Engineering, The University of Texas at San Antonio, San Antonio, TX, USA. Her research interests include security and privacy, machine learning, wireless networks, and Internet of Things. She is a recipient of the NSF CAREER Award, NSF/CISE CRII Award, Cisco Research Award, IEEE Computer Society TCSC Early Career Researchers Award for Excellence in Scalable Computing, and Rising Star in Networking and Communications Award by IEEE ComSoc N2Women. She is currently an associate editor of ACM Computing Surveys and IEEE Wireless Communications.