Privacy for Free: Communication-Efficient Learning with Differential Privacy Using Sketches

Tian Li
CMU
tianli@cmu.edu

Zaoxing Liu
CMU
zaoxing@cmu.edu

Vyas Sekar
CMU
vsekar@andrew.cmu.edu

Virginia Smith
CMU
smithv@cmu.edu

Abstract

Communication and privacy are two critical concerns in distributed learning. Many existing works treat these concerns separately. In this work, we argue that a natural connection exists between methods for communication reduction and privacy preservation in the context of distributed machine learning. In particular, we prove that Count Sketch, a simple method for data stream summarization, has inherent differential privacy properties.[1] Using these derived privacy guarantees, we propose a novel sketch-based framework (`DiffSketch`) for distributed learning, where we compress the transmitted messages via sketches to *simultaneously* achieve communication efficiency and provable privacy benefits. Our evaluation demonstrates that `DiffSketch` can provide strong differential privacy guarantees (e.g., $\varepsilon = 1$) and reduce communication by 20-50$\times$ with only marginal decreases in accuracy. Compared to baselines that treat privacy and communication separately, `DiffSketch` improves absolute test accuracy by 5%-50% while offering the same privacy guarantees and communication compression.

1 Introduction

Communication costs and privacy concerns are two critical challenges in performing distributed machine learning with sensitive information. Indeed, these challenges are particularly relevant for the increasingly common problem of *federated learning*, in which data is collected and stored across a network of devices such as mobile phones or wearable devices [29, 34]. Communicating data across such a distributed network in order to learn an aggregate model is both costly and can potentially reveal sensitive user information [10, 35].

Many prior efforts have separately considered communication or privacy in distributed machine learning. For example, common strategies to reduce the size of messages sent across the network include sparsification, quantization, or subsampling [25]. Previous privacy-preserving methods typically add noise to guarantee differential privacy [35], or use cryptographic protocols such as secure multi-party computation (SMC) [6]. While these approaches are effective at either reducing communication or protecting privacy independently, the two goals are largely treated as orthogonal to one another. As we demonstrate, this can be problematic when aiming to achieve both private and efficient learning, as using a direct combination of the above techniques [e.g., 3] can significantly degrade overall accuracy (§5.2).

In this work, we argue that the goals of communication reduction and privacy protection are closely related, and show that leveraging this relation has significant benefits for distributed learning. Intuitively, both tasks

[1] *We note that the definition of local differential privacy used in our work is weaker than the standard local privacy definition. In addition, we are aware of some issues with our current proof of the differential privacy properties of Count Sketch. We are currently working on a revision of this draft.*

share an aim to reduce, mask, or transform information sent across the network in a way that preserves the underlying learning task. While this connection has been observed in other settings [45, 47], we are not aware of other work that formalizes these connections and optimizes both goals in the context of distributed machine learning.

In particular, we identify *sketching algorithms* (sketches) as a natural tool to jointly consider communication and privacy in distributed learning. Sketches use independent hash functions to compress the input data with bounded errors, and have well-studied trade-offs between space and accuracy [4, 11, 14]. While sketches have previously been used to reduce communication in distributed networks [23], we show that by randomizing the data with independent hash functions, sketches can in fact also provide *privacy for free*, as they have provable differential privacy benefits (§3). More specifically, we demonstrate that sketches can be used to achieve *local differential privacy* [17, 44] in distributed machine learning—a setting where we do not assume a trusted central server and the central server cannot see individual model updates.

Based on our derived privacy guarantees, we introduce `DiffSketch`, a framework built upon canonical sketches such as Count Sketch [11] to jointly optimize communication and privacy in distributed learning. While we explore `DiffSketch` in the context of distributed machine learning, we note that it is general in that it could potentially be applied to any task that requires estimating the mean of a set of distributed vectors. Empirically, we verify the communication reduction and privacy benefits of `DiffSketch` for applications with both distributed SGD and `FedAvg`, a popular method for federated learning [34]. Our results demonstrate that by considering communication and privacy jointly, `DiffSketch` can deliver 5%-50% improvements in terms of absolute test accuracy compared with baselines for fixed levels of privacy and communication.

Contributions[2]**.** In this work, we provide the first differential privacy guarantees of Count Sketch without additional mechanisms (§3). Based on these derived privacy guarantees, we design `DiffSketch`, a communication-efficient and differentially-private distributed learning framework using Count Sketch as a building block (§4). Our evaluation demonstrates that `DiffSketch` offers strong privacy guarantees and communication reduction benefits with significantly improved accuracy compared to baselines (§5).

2 Related Work and Background

Communication-efficient learning. Communication is a key bottleneck in distributed learning, and has been studied extensively in classical distributed (e.g., data center) computing as well as emerging applications of federated learning. A common approach to reducing communication is to limit the size of messages sent across the network using various compression methods [e.g., 8, 23, 25]. This is particularly useful for applications such as deep learning, where the messages sent are similar in size to the model and can thus be quite large. As we show in this work, compression methods such as sketching are related to the goals of privacy preservation, but are typically explored in isolation.

A second, orthogonal approach to communication-efficiency is to reduce the total number of communication rounds in training by developing flexible and efficient distributed optimization methods [34, 39, 40, 41, 43]. While not the focus of this work, we explore our framework in conjunction with one such canonical communication-efficient optimization method, `FedAvg` [34], in §4.

[2]We note that a preliminary version of our vision appeared in [33], where we suggested the promise of using sketching to enhance privacy in federated learning. However, in this preliminary work we did not prove any differential privacy guarantees of sketches, propose a distributed learning framework to capitalize on the connection, or explore the privacy benefits empirically.

Privacy in distributed learning. In a variety of distributed learning scenarios handling user-generated data, privacy has become increasingly important to consider. Indeed, privacy concerns can even motivate the need for distributed learning—for example, privacy is a key motivation in federated learning, as it necessitates that raw user data remain local and thus distributed across the network [34]. There are two common approaches to preserve privacy in distributed settings, based either on statistical or cryptographical foundations. The first is to provide differential privacy by adding random perturbations (e.g., random noise drawn from a Gaussian or Laplacian distribution) to the output of a function [2, 3, 27, 35]. In distributed contexts, it is common to think of differential privacy as either being enforced in a "local" or "global" sense depending on whether the server is a trusted party. We provide more formal definitions of differential privacy in §3.1 and demonstrate that our framework, `DiffSketch`, is able to achieve strong local differential privacy guarantees via sketches.

Another line of work is based on cryptography, including secure multiparty computation (SMC) [6, 12, 21]. SMC protects users' privacy by encrypting individual inputs such that multiple parties can jointly compute a function without learning about the input information from any party. However, cryptographic-based approaches are not well-suited to distributed learning, as they can incur significant communication and computation overheads [5]. In general, these types of privacy-preserving methods tend to increase the communication cost in distributed settings.

Connections between communication and privacy. Some recent works have explored connections between communication compression and privacy [45, 47]. However, they focus on much simpler settings such as multiplicative database transformation [47], or transmitting raw data in sensor networks [45], and can only preserve limited information such as the covariance of the original data [47]. Duchi and Rogers [16] explore an equivalence between estimation under privacy constraints and estimation under communication constraints, but for different purposes of developing lower bounds to showcase the fundamental limitations of differential privacy. We explore such connections in distributed learning both theoretically and empirically through sketching, and demonstrate that `DiffSketch` achieves competitive accuracy with guaranteed differential privacy and convergence (§5). Finally, recent work called `cpSGD` [3] proposes modifications to distributed SGD to make the method both private and communication-efficient. However, the authors treat these as separate issues, and develop different approaches to address each within `cpSGD`. In §5 we compare directly with `cpSGD` and demonstrate that by *separately* reducing communication and enforcing privacy, errors in `cpSGD` are compounded; by instead considering these *jointly*, `DiffSketch` provides substantially higher accuracy.

Sketches. Sketching algorithms (sketches) provide approximate estimates of different statistics (such as computing the distinct count or average) over a dataset, and have been widely used in streaming data processing [4, 15, 37], databases [11, 13, 14], and network measurement [31, 32, 46]. Recently sketches have also been used to improve large-scale machine learning, e.g., by compressing model updates [24], identifying significant coordinates in the gradient vectors [23], and reducing memory usage in model training [42]. However, these works do not focus on privacy.

Some recent efforts consider extensions to sketches to provide differential privacy through the use of random noise [36], random sampling [48], or other randomization [5, 21]. However, they require adding these mechanisms on top of sketches, and do not investigate any inherent privacy properties of sketches themselves. Our analysis shows that sketches have inherent differential privacy properties (§3) and can be utilized to design private distributed learning algorithms that achieve better communication and accuracy trade-offs than state-of-the-art mechanisms (§4).

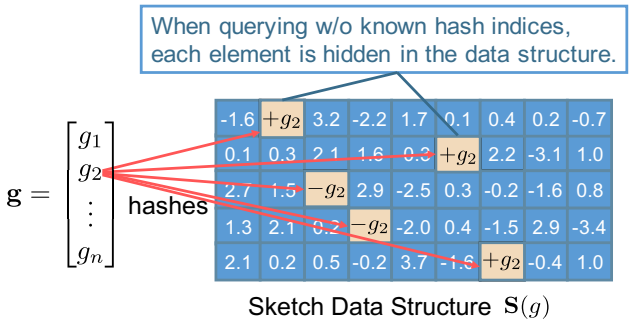

Figure 1: An illustration of using Count Sketch $\boldsymbol{S}_{5\times 9}$ to compress a vector $g \in \mathbb{R}^n$. Each element $g_i \in g$ is mapped to five bins in five counter arrays via independent hash functions. It is difficult for third parties to infer the original inputs. We prove that the Count Sketch algorithm is differentially private in §3.2.

3 Differential Privacy of Sketches

In this section, we analyze the intrinsic differential privacy properties of a canonical sketch (Count Sketch). We begin by providing relevant background on Count Sketch and differential privacy, and then present our differential privacy bounds in §3.2.

3.1 Preliminaries

Count Sketch is a common sketching algorithm that can be used to compress real-valued vectors. As depicted in Figure 1, Count Sketch works by using t independent hash functions to map each element in a vector $g \in \mathbb{R}^n$ to t distinct bins in arrays of size $k \ll n$. This may cause some collisions, and the goal of the sketch is therefore to approximate the true values within bounded errors when queried. Due to the linearity of Count Sketch, it can be used to compress messages in distributed settings, as it is easy to merge sketched messages since $\boldsymbol{S}(g_1 + g_2) = \boldsymbol{S}(g_1) + \boldsymbol{S}(g_2)$ for a sketch method $\boldsymbol{S}$ and vectors g_1 and g_2.

We summarize Count Sketch in Algorithm 1, and defer interested readers to Charikar et al. [11] for additional background. The algorithm consists of two key operations for any vector $g \in \mathbb{R}^n$:

1. *Encoding g into a sketch table $\boldsymbol{S}_{t\times k}(g)$.* To encode every $g_i \in g$, we use a set of pairwise independent hash functions $\{h_{j,1\le j\le t} : [n] \to [k]\}$, along with a set of 2-wise independent sign hash functions $\{\text{sign}_{j,1\le j\le t} : [n] \to \{+1, -1\}\}$ to map each g_i to t different bins in the arrays of the table.
2. *Querying $\boldsymbol{S}_{t\times k}(g)$ to obtain an estimation $\tilde{g}$ of g.* To query g_i from $\boldsymbol{S}_{t\times k}(g)$, we query the median of t approximated values identified by the indexes of $h_j(i)$ $(1 \le j \le t)$. To achieve μL_2 additive errors on $\tilde{g}$ with $1-\delta$ probability, the size of the table (t and k) are configurable as $t = O\left(\ln(\frac{1}{\delta})\right)$ and $k = O\left(\frac{1}{\mu^2}\right)$ [11].

When analyzing Count Sketch, our insight is that the randomization from the independent hash functions potentially provides differential privacy in the sketch table. Intuitively, in Count Sketch, the differences in the output distribution caused by minor changes in the input should be bounded with high probability. This observation mirrors the definition of differential privacy, as formally stated below.

Algorithm 1 Count Sketch to compress $g \in \mathbb{R}^n$.

Input: $g \in \mathbb{R}^n$, t, k, $\boldsymbol{S}_{t\times k}$, h_i $(1 \le i \le t)$, sign_i $(1 \le i \le t)$

Compress vector $g \in \mathbb{R}^n$ into $\boldsymbol{S}(g)$:
Initialize $\boldsymbol{S}_{t\times k}$ to all zeros's
for $g_i \in g$ **do**
  for $j = 1, \cdots, t$ **do**
    $\boldsymbol{S}[j][h_j(i)] \leftarrow \boldsymbol{S}[j][h_j(i)] + \text{sign}_j(i) \cdot g_i$
  end for
end for
return $\boldsymbol{S}_{t\times k}$

Query $\tilde{g} \in \mathbb{R}^n$ from $\boldsymbol{S}(g)$:
for $i = 1, \cdots, n$ **do**
  $\tilde{g_i} = \text{Median}\{\text{sign}_j(i) \cdot \boldsymbol{S}[j][h_j(i)] : 1 \le j \le t\}$
end for
return $\tilde{g}$

Definition 1 (*ε-differential privacy* [18]). A randomized mechanism $\mathcal{M}$ satisfies ε-differential privacy, if for input data D_1 and D_2 differing by up to one element, and for any output S of $\mathcal{M}$,

$$Pr[\mathcal{M}(D_1) \in S] \le e^{\varepsilon} \cdot Pr[\mathcal{M}(D_2) \in S].$$

Informally, the above definition states that when ε is small enough, the two output distributions are closer to each other, and the method $\mathcal{M}$ becomes more private as it is difficult to determine whether the input is D_1 or D_2. There are two common privacy models in distributed networks: *local privacy* where the central aggregator/server is not trusted, and *global privacy* where we assume a trusted central server [17, 44]. As discussed in §4, our proposed framework `DiffSketch` leveraging Count Sketch satisfies the stronger local privacy model. We next analyze the differential privacy properties of Count Sketch.

3.2 Differential Privacy Analysis

To analyze the privacy guarantees of sketches, we need to provide some measurement of the output distribution that reflects the amount of noise that results from hash collisions in the sketch. To achieve this, we quantify the input data distribution with the following assumptions: (a) each element in the input vector is bounded with a large probability, and (b) the inputs are drawn from Gaussian distributions. Under these assumptions, we provide the first differential privacy guarantees of Count Sketch without any additional noise.

Assumption 1 (Input vector distribution). *Following the work of Fergus et al. [19], Gong and Sbalzarini [22], Levin et al. [26], Parmas [38], we assume that for any input vector D with length $|D| = n$, each element $d_i \in D$ is drawn I.I.D. from a Gaussian distribution: $d_i \sim \mathcal{N}(0, \sigma^2)$, and bounded by a constant with a large probability: $|d_i| \le \alpha, 1 \le i \le n$, for some constant α.*

As discussed in §4, we apply Count Sketch to compress gradients (or model updates) in distributed learning. Therefore, the input vectors are gradients (or model updates) generated from the workers' local data. We note that similar assumptions on normality have been previously used to characterize gradients for other applications [e.g., 19, 22, 26, 38]. We also empirically verify that the gradients or model updates closely match Gaussian distributions, and plot the distribution of real updates in Figure 7 and 8 in Appendix C.2. As we will discuss, our local privacy bound is a function of the local gradients/model updates bound α and

its variance σ^2. When leveraging sketches for privacy, we dynamically estimate α and σ^2 locally for each participating worker during training without prior knowledge, and use the estimated values to compute the privacy parameter ε.

Next, we state a lemma about Count Sketch's error bounds before proving the differential privacy in our main theorem.

Lemma 1 (Estimation error of Count Sketch [11]). *For a sketching mechanism $\mathcal{M}$ using Count Sketch with t arrays of k bins, for any input element $d_i \in D$ and query $\mathcal{Q}$, with probability $p \geq 1 - \delta$,*

$$|\mathcal{Q}(\mathcal{M}(d_i)) - d_i| \leq \mu \|D\|_2,$$

where $k = O\left(\frac{e}{\mu^2}\right)$, and $t = O\left(\ln(\frac{1}{\delta})\right)$.

Lemma 1 states the error guarantee of Count Sketch, i.e., that with high probability, the recovered/estimated values are very close to the original value, with a small error up to a fraction μ of L_2 norm of the input vector. With this lemma, we can translate the bound of the input values to the bound of every value in the output sketching table, which will be used in our proof. We provide the differential privacy guarantees of a basic Count Sketch $\mathcal{M}$ in our main theorem below.

Theorem 2 (ε-differential privacy of Count Sketch). *For a sketching algorithm $\mathcal{M}$ using Count Sketch $\boldsymbol{S}_{t\times k}$ with t arrays of k bins, for any input vector D with length n satisfying Assumption 1, $\mathcal{M}$ achieves $t \cdot \ln\left(1 + \frac{\beta\alpha^2 k(k-1)}{\sigma^2(n-2)}\left(1 + \ln(n-k)\right)\right)$-differential privacy with high probability, where β is a positive constant satisfying $\frac{\alpha^2 k(k-1)}{\sigma^2(n-2)}\left(1 + \ln(n-k)\right) \leq \frac{1}{2} - \frac{1}{\beta}$.*

We defer readers to Appendix A for a detailed proof, and provide a high-level sketch here: Since the t counter arrays are independent, we first derive the output probability density functions of a single counter array. Based on the sketching algorithm (Algorithm 1) as well as the input distributions (Assumption 1), we can show that the output distribution is a mixture of Gaussians. Since we assume uniformly bounded inputs, we can then prove that the difference of two output distributions before and after one input element changes is also bounded. The overall privacy guarantee is then multiplied by t given the independence between arrays.

To the best of our knowledge, we provide the first differential privacy proof for vanilla Count Sketch without additional randomization. From Theorem 2, we see that the privacy parameter ε will become smaller when the size of the input vector, n, gets smaller, and we can achieve nearly zero-differential privacy when n is large enough. Theorem 2 also reveals natural trade-offs between accuracy, communication, and privacy. For instance, if we use a smaller sketching table to compress the input vector (smaller t or k), we are compressing more aggressively (losing more information) and $\mathcal{M}$ will be more private with a smaller ε parameter—potentially at the cost of overall accuracy.

4 Proposed Framework: `DiffSketch`

In this section, we introduce `DiffSketch`, a general framework for distributed learning. We begin by describing `DiffSketch`'s end-to-end approach for reducing communication and enforcing differential privacy using sketches (§4.1). We then show how `DiffSketch` can be applied to both classical distributed machine learning, with distributed SGD (§4.2), and federated learning, with `FedAvg` (§4.3).

Algorithm 2 Proposed framework: `DiffSketch`.

Input: T, $v_1, \cdots, v_m$, ε
for $t = 0, \cdots, T-1$ **do**
  Compression: Each entity k compresses v_k to obtain $\boldsymbol{S}(v_k)$ based on Algorithm 1
  Validation: Each entity validates if it satisfies ε-differential privacy for a given ε. If not, add appropriate Laplacian or Gaussian noise to $\boldsymbol{S}(v_k)$
  Aggregation: The server aggregates local information to obtain $\boldsymbol{S}(v) = \frac{1}{m}(\boldsymbol{S}_1 + \cdots + \boldsymbol{S}_m)$
  Query: Each entity queries $\boldsymbol{S}(v)$ for the mean $\tilde{v}$ based on Algorithm 1
end for

4.1 `DiffSketch`: A Framework for Distributed Learning

`DiffSketch` is an efficient and private framework for distributed learning. On top of the existing distributed methods, `DiffSketch` uses sketches to both preserve the privacy and reduce the size of transmitted messages with small overhead. In particular, many distributed optimization methods perform work in parallel (e.g., computing gradients), and then aggregate local information from distributed entities by calculating a mean (e.g., computing the average mini-batch of gradients). These tasks can be completed using a sketch through the *Aggregation* and *Query* steps. As depicted in Algorithm 2, `DiffSketch` has two additional simple operations: *Compression* and *Validation*.

The complete operations in `DiffSketch` are as follows: *Compression:* At each round, entity k uses sketches to compress the local information v into a privatized $\boldsymbol{S}(v_k)$. *Validation:* Once the compression is done, each local worker needs to verify if sketching alone satisfies ε-differential privacy (based on Theorem 2). If not, we may need to ensure a consistent privacy guarantee with some additional noise. As sketches have provable privacy, zero or a small amount of noise is expected (we verify this empirically in §5). *Aggregation:* The central server aggregates the local information and sends the aggregated (i.e., averaged) local updates back. *Query:* Each local entity approximately recovers the updated global information from the (merged) sketch.

As mentioned in the related work (§2) and preliminaries (§3.1), local differential privacy does not assume a trusted central server, and the server cannot see individual inputs. Since `DiffSketch` allows each worker to sketch the vectors locally before sending the compressed sketching table to the server, we are privatizing data at a local level, thus protecting against third-parties including the central server.

Generality. We note that `DiffSketch` is a general framework for a variety of distributed learning scenarios where user data needs to be protected. In this paper, we describe the applications to distributed SGD (§4.2) and federated learning (§4.3). It can also be potentially applied to broader distributed data analysis tasks with privacy concerns; the main requirement is simply that the workload is aggregating information across the network via calculation of a mean of distributed vectors. `DiffSketch`'s modular design has two major advantages: First, the provable accuracy guarantees and differential privacy for the compressed data help any analysis algorithms to retain high accuracy. Second, the simple hashing-based computations in applying the sketches to Compression and Query are light-weight operations. One can leverage efficient CPU parallelism or hardware acceleration (e.g., SSE and AVX) to achieve optimized computation performance.

4.2 DiffSketch for Distributed SGD

We first explore DiffSketch with distributed (mini-batch) SGD as a subroutine. Throughout the paper, we assume the widely-used objective of minimizing the finite sum of the empirical loss:

$$\min_{w} F(w) = \sum_{k=1}^{m} p_k F_k(w)\,, \tag{1}$$

where F_k is the local empirical loss on worker k, m is the total number of workers, and p_k is the weight set for worker k (e.g., $\frac{1}{m}$). We consider possibly differing local data distributions across the network and possibly non-convex F_k's. In this work, we assume a classical synchronous and centralized training setup with m workers connected to one central server.

A typical workflow of using distributed SGD to solve (1) is that at each updating round, each local worker computes gradients using (a mini-batch of) local data and sends the gradients to the server, where they get merged and sent back to the workers. We can directly apply Algorithm 2 to compress the gradients g's sent from workers to the server. The server can not see the raw gradients as they have been compressed and masked into small sketching tables. See Algorithm 3 for more details. As sketches are guaranteed to preserve the original skewed gradient values with high probabilities, the local workers can recover the merged gradients with very high accuracies. In our experiments (§5), we demonstrate that the accuracy reduction is very minor while the compression ratio is high (up to 50×).

Convergence. We also provide convergence guarantees for DiffSketch with distributed SGD for convex functions. Our convergence results rely on the bounded estimation error of Count Sketch (Lemma 1), and unbiasedness of Count Sketch [11]: For a Count Sketch $\mathcal{M}$, for any input element $d_i \in D$ and query $\mathcal{Q}$, $\mathbb{E}\left[\mathcal{Q}(\mathcal{M}(d_i)\right] = d_i$. Together with Lemma 1, this guarantees that the estimated gradients are both unbiased and uniformly bounded with high probability.

Theorem 3 (Convergence of DiffSketch in distributed SGD)**.** *Assume that $E\left[\|g_k\|^2\right] \leq G^2$ for any input stochastic gradient g_k on device k with dimension n, $E\left[\|x_i - x^*\|^2\right] \leq D^2$ at any iteration i, and the local model at device k $f_k(x)$ is convex. Choose the step-size at round i $\eta_i = \frac{c}{\sqrt{i}}$ where c is a pre-defined positive number. Using a Count Sketch with t hash functions and k bins, with probability $(1-\delta)$, we have:*

$$F(\bar{x}_i) - F(x^*) \leq \frac{\frac{D^2}{2c} + c\sqrt{\frac{i+1}{i}}\left(n\mu^2 + 1\right)G^2}{\sqrt{i}},$$

where x^* is the optimal solution to (1), $k = O\left(\frac{e}{\mu^2}\right)$, $t = O\left(\ln(\frac{1}{\delta})\right)$, and $\bar{x}_i = \frac{1}{i}\sum_{j=1}^{i} x_j$.

For simplicity, we abuse notations and use $g_k \in \mathbb{R}^n$ to denote the gradient vector on device k. We provide a detailed proof in Appendix B. Theorem 3 indicates that DiffSketch has the same convergence rate as standard distributed SGD [20] under convex settings. It also indicates that as the number of hash functions t and the number of bins k increase, we compress less and tend to get a tighter convergence bound (due to a smaller recovery error μ) with a higher probability (lower δ).

4.3 DiffSketch for Federated Learning

Federated learning aims to fit a model to data generated by, and residing on, networks of hundreds to millions of remote devices [34]. Applying DiffSketch must carefully consider unique challenges associated with this

Algorithm 3 `DiffSketch` with distributed SGD.

Input: T, η, w^0, ε
for $t = 0, \cdots, T-1$ **do**
  if $t > 0$ **then**
    Server sends the sketched global gradient $\boldsymbol{S}(g^t)$ to all workers
    Each worker queries $\boldsymbol{S}(g^t)$ for $\tilde{g}^t$
    Each worker updates: $w^t = w^{t-1} - \eta \tilde{g}^t$
  end if
  Each worker k runs (mini-batch) SGD on w^t to obtain local gradients g_k^{t+1}
  Each worker sketches the gradients locally to obtain $\boldsymbol{S}(g_k^{t+1})$
  Each worker adds additional Laplacian noise to $\boldsymbol{S}(g_k^{t+1})$ if not satisfying ε-differential privacy
  Each worker sends $\boldsymbol{S}(g_k^{t+1})$ to the server
  Server aggregates the model updates: $\boldsymbol{S}(g^{t+1}) = \frac{1}{m}\sum_{k=1}^{m} \boldsymbol{S}(g_k^{t+1})$
end for

gradient.

Algorithm 4 `DiffSketch` in federated learning.

Input: K, T, η, E, w^0, p_k, ε
for $t = 0, \cdots, T-1$ **do**
  Server samples a subset S_t of K devices (each device is chosen with probability p_k)
  if $t > 0$ **then**
    Server sends the sketched global model $\boldsymbol{S}(\Delta w^t)$ to all chosen devices
    Each device k queries $\boldsymbol{S}(\Delta w^t)$ for $\Delta \tilde{w}^t$
    Each device k updates: $w^t = w^{t-1} + \Delta w^t$
  end if
  Each device k updates w^t for E epochs of SGD on F_k with step-size η to obtain Δw_k^{t+1}
  Each device k sketches the updates locally to obtain $\boldsymbol{S}(\Delta w_k^{t+1})$
  Each device k adds additional Laplacian noise to $\boldsymbol{S}(\Delta w_k^{t+1})$ if not satisfying ε-differential privacy
  Each device k sends $\boldsymbol{S}(\Delta w_k^{t+1})$ to the server
  Server aggregates the model updates: $\boldsymbol{S}(\Delta w^{t+1}) = \frac{1}{K}\sum_{k \in S_t} \boldsymbol{S}(\Delta w_k^{t+1})$
end for

updates

setting—the large scale of the networks, expensive communication, strict privacy requirements, and a high degree of heterogeneity across devices [29]. In practice, it is common that only a small fraction of devices are active at each round [7]. Optimization methods using local updating and tolerating low participation of devices have become the de facto solvers for federated settings; of these, `FedAvg` is most widely-used [34].

`DiffSketch` can use `FedAvg` as a subroutine and similarly account for important characteristics in federated learning. At each communication round, it randomly samples a subset of devices, lets each participating device perform E epochs of local updates, applies Count Sketch to compress the updates, and averages the updates centrally. Details are summarized in Algorithm 4. Empirically, we demonstrate that `DiffSketch` can compress communication by up to 20× and provide strong local privacy guarantees ($\varepsilon = 1$) on real federated datasets (§5).

We note that in federated settings with non-identically distributed data, `FedAvg` is a heuristic and may not converge despite its overall robust practical performance [28, 34]. Therefore, we also do not provide convergence guarantees for `DiffSketch` in this setting, though we explore the method empirically in §5.

FedAvg 即使 robuse 性能，但可能不会收敛.

5 Evaluation

We now present empirical results for the `DiffSketch` framework. In §5.2, we compare `DiffSketch` with other baselines that can reduce communication and preserve privacy simultaneously, and demonstrate the superior performance of `DiffSketch`. In §5.3, we investigate the trade-off between privacy, communication, and accuracy in `DiffSketch`. All code, data, and experiments are publicly available at `github.com/litian96/DiffSketch`.

5.1 Simulation Setups

Datasets. For distributed SGD, we randomly subsample from the MNIST dataset for image classification and form 10 partitions with identical distributions across local workers. For federated learning, we use the Shakespeare dataset, which is a popular dataset curated from federated learning benchmarks [9]. It is naturally partitioned in a heterogeneous way where each device is associated with a speaking role in the plays. To investigate a setup that is favorable to `cpSGD` [3], we partition MNIST into 6,000 subsets with one subset corresponding to a device, as we observe that the privacy bound in `cpSGD` becomes tighter (better) if sampling from a larger number of devices. Data statistics are summarized in Table 2, Appendix C.

Implementation. We implement Algorithm 3 and 4 in Tensorflow [1], and simulate a one server and m workers setup (see Table 2 in Appendix C). To further improve the performance of sketches, we consider several simple implementation optimizations as follows. In distributed SGD, each local worker compares the queried results with the local gradients, and sets half of the gradients with inaccurate estimations (larger gaps) to zero. For both distributed SGD and federated learning, we generate some random noise drawn from the same Gaussian distribution as the original gradients (or model updates), and append the noise vector after the input. By increasing the size of the input and thus compressing more values, we are boosting the privacy of sketches. All ε values reported in the following experiments are local privacy guarantees—each local worker achieves ε-differential privacy at each round. See full details in Appendix C.1.

5.2 Comparison with Baselines

In this section, we compare with several baseline methods that address both communication and privacy, including `cpSGD` [3], a state-of-the-art method that aims optimize both aspects (though it treats each separately). As secure multi-party computation (SMC) is generally inefficient in distributed learning (as discussed in §2), we do not compare with methods based on SMC.

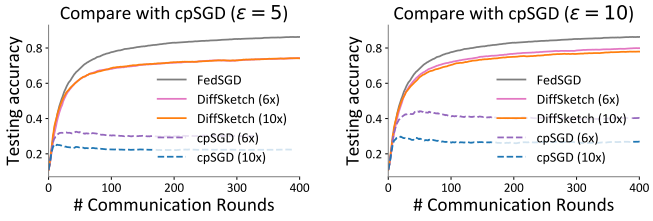

Figure 2: `DiffSketch` compared with `cpSGD`. Under the same privacy guarantees ($\varepsilon = 5$ or 10) and the same communication compression ratio, `DiffSketch` achieves significantly higher accuracy. If we continue to decrease ε to improve the privacy, `cpSGD` will perform even worse.

cpSGD. `cpSGD` [3] first quantizes the gradients to reduce communication, then adds Binomial noise to the quantized gradients to offer differential privacy. We compare `DiffSketch` with `cpSGD` on MNIST*. Results are shown in Figure 2. We see that across different ε values and compression ratios, by considering privacy and communication jointly, `DiffSketch` achieves significant improvements of up to 50% in *absolute* test accuracy over `cpSGD`.

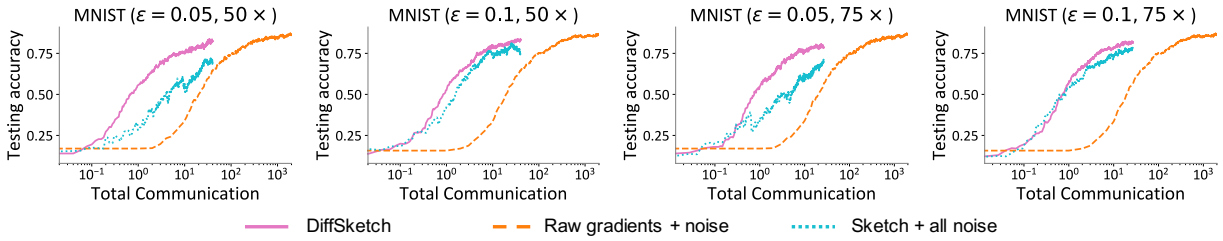

Figure 3: `DiffSketch` compared with other baselines in the distributed SGD setting. We show the test accuracy versus total communication in log scale. The total amount of communication is normalized via dividing the communication rounds by the compression ratio. Given the same amount of privacy, (1) `DiffSketch` converges with orders-of-magnitude less communication than directly adding noise to the gradients (orange line), and (2) `DiffSketch` is more accurate than treating sketches as plain-text and directly adding noise to sketches at all communication rounds (blue line). This indicates that `DiffSketch` is inherently private such that less (or zero) additional noise is sufficient to provide the same privacy guarantees. The sketch sizes of $50\times$ and $75\times$ compression ratios are 7×22 and 7×15, respectively.

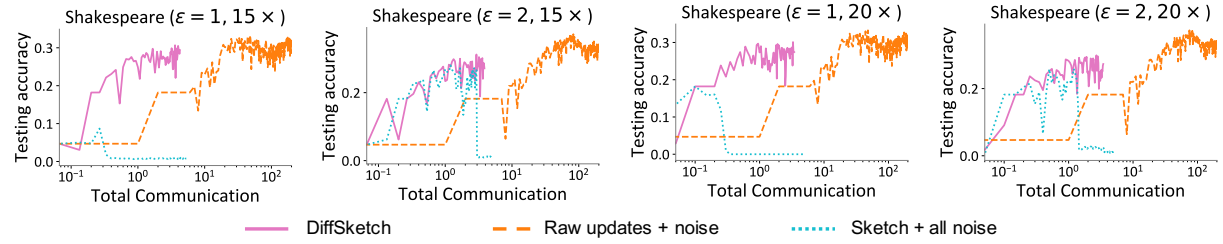

Figure 4: `DiffSketch` compared with other baselines in federated learning. Again, we note here that the total amount of communication is in log scale. Similar to Figure 3, it indicates that `DiffSketch` can significantly improve test accuracy while offering the same privacy and communication benefits. We observe that smaller ε values would cause baseline methods to diverge, and so we investigate slightly larger ε's. The sketch sizes of $15\times$ and $20\times$ compression ratios are 10×190 and 10×245.

Other baselines. We next compare with two natural alternatives that optimize for privacy in practice: (1) directly adding Laplacian noise to the raw gradients transmitted between the workers and the central server to offer (local) differential privacy, and (2) treating sketches as plain-text, and adding Laplacian noise after sketching at all updating rounds to offer both privacy and compression [36]. The results are shown in Figure 3 (for distributed SGD) and Figure 4 (for federated learning). In both settings, we see that `DiffSketch` converges faster than adding noise to raw gradients as the amount of communication is significantly reduced by sketching. Also, `DiffSketch` is more accurate than adding additional noise to sketches at all iterations, especially when ε is smaller. This is because in order to guarantee stronger privacy with smaller ε's, the baseline approach (in blue) needs to add more noise on top of sketches, thus hurting model performance, while `DiffSketch` only adds a small amount of noise when necessary (Figure 9 in the appendix).

5.3 Trade-offs in `DiffSketch`

Finally, we explore trade-offs in `DiffSketch` between communication, privacy, and accuracy.

Accuracy vs. compression. We plot the testing accuracy with the total amount of communication under different compression ratios for both distributed SGD and federated learning in Figure 5. As the compression ratio increases, we obtain faster convergence with less communication, but have potentially lower final accuracy. We note that many existing works in model compression explore techniques for improving accuracy under high compression ratios [e.g., 25, 30]; although outside the focus of this work, these techniques could be combined with our work to further boost the compression ratios.

Privacy vs. communication. We show how compression relates to differential privacy in Figure 6. The ε values are averaged across all communication rounds. In our experiments, sketches themselves are sufficient to provide privacy benefits without additional noise in most communication rounds. We can see that a higher compression ratio leads to more privacy (smaller ε).

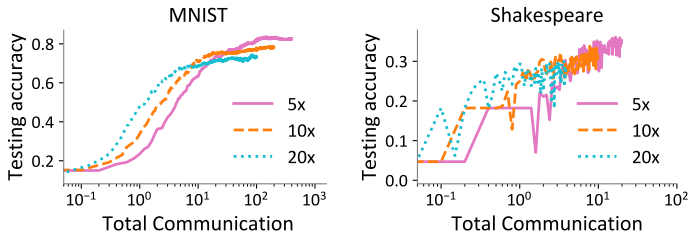

Figure 5: As the compression ratio becomes higher, `DiffSketch` converges faster, but with potentially lower accuracies.

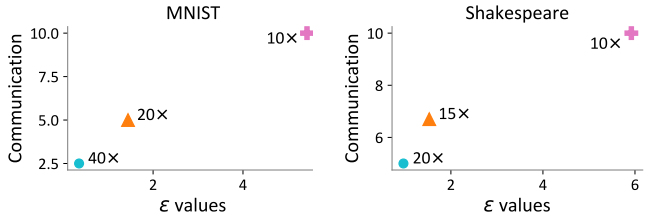

Figure 6: As we compress more, we can reduce communication and obtain stronger privacy guarantees with smaller ε values.

6 Conclusion

In this work, we examined connections between communication reduction and privacy preservation in distributed machine learning. We first proved that canonical sketches (Count Sketch) have inherent differential privacy benefits without additional mechanisms. Based on our theoretical understandings, we designed `DiffSketch`, a framework for efficient and private distributed learning. We applied `DiffSketch` to classical distributed SGD and federated learning, and demonstrated empirically that `DiffSketch` achieves 5%-50% absolute accuracy improvements compared with baselines while offering the same communication and privacy benefits. While we explored such connections via sketches, a natural direction of future work is to investigate whether other tools from the privacy or distributed learning communities may similarly provide benefits for both privacy and communication simultaneously.

Acknowledgement

We thank Jalaj Upadhyay and Weizhao Tang for their helpful discussions. This work was supported in part by the National Science Foundation grant IIS1838017, CNS-1700521, CNS-1565343, a Google Faculty Award, a Carnegie Bosch Institute Research Award, Intel Labs University Research Office, and the CONIX Research Center. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the National Science Foundation or any other funding agency.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. K. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. Tensorflow: A system for large-scale machine learning. In *Operating Systems Design and Implementation*, 2016.

[2] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *Conference on Computer and Communications Security*, 2016.

[3] N. Agarwal, A. T. Suresh, F. X. X. Yu, S. Kumar, and B. McMahan. cpSGD: Communication-efficient and differentially-private distributed sgd. In *Advances in Neural Information Processing Systems*, 2018.

[4] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. In *Symposium on Theory of Computing*, 1996.

[5] R. Bassily, K. Nissim, U. Stemmer, and A. G. Thakurta. Practical locally private heavy hitters. In *Advances in Neural Information Processing Systems*, 2017.

[6] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth. Practical secure aggregation for privacy-preserving machine learning. In *Conference on Computer and Communications Security*, 2017.

[7] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecny, S. Mazzocchi, H. B. McMahan, T. V. Overveldt, D. Petrou, D. Ramage, and J. Roselander. Towards federated learning at scale: system design. In *Conference on Systems and Machine Learning*, 2019.

[8] S. Caldas, J. Konečny, H. B. McMahan, and A. Talwalkar. Expanding the reach of federated learning by reducing client resource requirements. *arXiv preprint arXiv:1812.07210*, 2018.

[9] S. Caldas, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

[10] N. Carlini, C. Liu, Ú. Erlingsson, J. Kos, and D. Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *USENIX Security Symposium*, 2019.

[11] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. In *International Colloquium on Automata, Languages, and Programming*, 2002.

[12] V. Chen, V. Pastro, and M. Raykova. Secure computation for machine learning with spdz. *arXiv preprint arXiv:1901.00329*, 2019.

[13] G. Cormode. Sketch techniques for approximate query processing. *Foundations and Trends in Databases*, 2011.

[14] G. Cormode and S. Muthukrishnan. An Improved Data Stream Summary: The Count-Min Sketch and Its Applications. *Journal of Algorithms*, 2005.

[15] G. Cormode, F. Korn, S. Muthukrishnan, and D. Srivastava. Finding hierarchical heavy hitters in streaming data. *Transactions on Knowledge Discovery from Data*, 2008.

[16] J. Duchi and R. Rogers. Lower bounds for locally private estimation via communication complexity. In *Conference on Learning Theory*, 2019.

[17] J. C. Duchi, M. I. Jordan, and M. J. Wainwright. Local privacy and statistical minimax rates. In *Foundations of Computer Science*, 2013.

[18] C. Dwork. Differential privacy. *Encyclopedia of Cryptography and Security*, 2011.

[19] R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis, and W. T. Freeman. Removing camera shake from a single photograph. In *ACM Transactions on Graphics*, 2006.

[20] S. Ghadimi and G. Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 2013.

[21] B. Ghazi, R. Pagh, and A. Velingker. Scalable and differentially private distributed aggregation in the shuffled model. *arXiv preprint arXiv:1906.08320*, 2019.

[22] Y. Gong and I. F. Sbalzarini. Gradient distribution priors for biomedical image processing. *arXiv preprint arXiv:1408.3300*, 2014.

[23] N. Ivkin, D. Rothchild, E. Ullah, V. Braverman, I. Stoica, and R. Arora. Communication-efficient distributed sgd with sketching. In *Advances in Neural Information Processing Systems*, 2019.

[24] J. Jiang, F. Fu, T. Yang, and B. Cui. Sketchml: accelerating distributed machine learning with data sketches. In *International Conference on Management of Data*, 2018.

[25] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon. Federated learning: strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[26] A. Levin, R. Fergus, F. Durand, and W. T. Freeman. Image and depth from a conventional camera with a coded aperture. *ACM Transactions on Graphics*, 2007.

[27] J. Li, M. Khodak, S. Caldas, and A. Talwalkar. Differentially private meta-learning. *arXiv preprint arXiv:1909.05830*, 2019.

[28] T. Li, A. K. Sahu, M. Sanjabi, M. Zaheer, A. Talwalkar, and V. Smith. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018.

[29] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith. Federated learning: Challenges, methods, and future directions. *arXiv preprint arXiv:1908.07873*, 2019.

[30] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. *International Conference on Learning Representations*, 2018.

[31] Z. Liu, A. Manousis, G. Vorsanger, V. Sekar, and V. Braverman. One sketch to rule them all: Rethinking network flow monitoring with univmon. In *Proceedings of the ACM Special Interest Group on Data Communication*, 2016.

[32] Z. Liu, R. Ben-Basat, G. Einziger, Y. Kassner, V. Braverman, R. Friedman, and V. Sekar. Nitrosketch: Robust and general sketch-based monitoring in software switches. In *Proceedings of the ACM Special Interest Group on Data Communication*, 2019.

[33] Z. Liu, T. Li, V. Smith, and V. Sekar. Enhancing the privacy of federated learning with sketching. 2019.

[34] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Conference on Artificial Intelligence and Statistics*, 2017.

[35] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang. Learning differentially private recurrent language models. In *International Conference on Learning Representations*, 2018.

[36] L. Melis, G. Danezis, and E. D. Cristofaro. Efficient private statistics with succinct sketches. In *Network and Distributed System Security Symposium*, 2016.

[37] A. Metwally, D. Agrawal, and A. E. Abbadi. Efficient computation of frequent and top-k elements in data streams. In *International Conference on Database Theory*, 2005.

[38] P. Parmas. Total stochastic gradient algorithms and applications in reinforcement learning. In *Advances in Neural Information Processing Systems*, 2018.

[39] S. J. Reddi, J. Konečnỳ, P. Richtárik, B. Póczós, and A. Smola. Aide: Fast and communication efficient distributed optimization. *arXiv preprint arXiv:1608.06879*, 2016.

[40] V. Smith, C.-K. Chiang, M. Sanjabi, and A. Talwalkar. Federated multi-task learning. In *Advances in Neural Information Processing Systems*, 2017.

[41] V. Smith, S. Forte, C. Ma, M. Takac, M. I. Jordan, and M. Jaggi. Cocoa: a general framework for communication-efficient distributed optimization. *Journal of Machine Learning Research*, 2018.

[42] R. Spring, A. Kyrillidis, V. Mohan, and A. Shrivastava. Compressing gradient optimizers via count-sketches. In *International Conference on Machine Learning*, 2019.

[43] S. U. Stich. Local sgd converges fast and communicates little. In *International Conference on Learning Representations*, 2019.

[44] S. L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 1965.

[45] S. Xiong, A. D. Sarwate, and N. B. Mandayam. Randomized requantization with local differential privacy. In *International Conference on Acoustics, Speech and Signal Processing*, 2016.

[46] M. Yu, L. Jose, and R. Miao. Software defined traffic measurement with opensketch. In *USENIX Symposium on Networked Systems Design and Implementation*, 2013.

[47] S. Zhou, K. Ligett, and L. Wasserman. Differential privacy with compression. In *International Symposium on Information Theory*, 2009.

[48] W. Zhu, P. Kairouz, H. Sun, B. McMahan, and W. Li. Federated heavy hitters discovery with differential privacy. *arXiv preprint arXiv:1902.08534*, 2019.

A Proof for Theorem 2

Theorem: (ε-differential privacy of Count Sketch) For a sketching algorithm $\mathcal{M}$ using Count Sketch $\boldsymbol{S}_{t\times k}$ with t arrays of k bins, for any input vector D with length n satisfying Assumption 1, $\mathcal{M}$ achieves $t\ln\left(1+\frac{\beta\alpha^2k(k-1)}{\sigma^2(n-2)}\left(1+\ln(n-k)\right)\right)$-differential privacy with high probability, where β is a positive constant satisfying $\frac{\alpha^2k(k-1)}{\sigma^2(n-2)}\left(1+\ln(n-k)\right)\leq\frac{1}{2}-\frac{1}{\beta}$.

Notations. We first list the notations that will be used throughout the proof.

- Input vector: $D=\{d_1,\cdots,d_n\},\ |D|=n$
- The constructed data structure: $\boldsymbol{S}$
- Number of bins in one counter array (i.e., one row of $\boldsymbol{S}$): k
- Number of hash functions (number of counter arrays): t
- Random variables of the bin values in one counter array: $B_1,\cdots,B_k$
- Bin values in one counter array: $b_1,\cdots,b_k$
- Random variables of numbers of elements in D mapped to k bins: $A_1,\cdots,A_k$
- Numbers of elements in D mapped to k bins: $a_1,\cdots,a_k$

We state one lemma that will be used in the proof for Theorem 2

Lemma 2. *Given* $\sum_{i=1}^k a_i=n-1$, $a_i\in\mathbb{Z}^+$, $a_1!\cdot a_2!\cdots a_k!\cdot\sqrt{a_1\cdot a_2\cdots a_k}$ *is minimized when* $\max\{|a_i-a_j|,1\leq i,j\leq k\}\leq 1$*; also* $\frac{1}{a_1}+\cdots+\frac{1}{a_k}$ *is minimized when* $\max\{|a_i-a_j|,1\leq i,j\leq k\}\leq 1$.

Proof. Let $f(a_1,a_2,\cdots,a_k)=a_1!\cdot a_2!\cdots a_k!\cdot\sqrt{a_1\cdot a_2\cdots a_k}$. If there exists $a_j\neq a_i$ and $a_i+2\leq a_j$, then $f(a_1,\cdots,a_i+1,\cdots,a_j-1,\cdots,a_k)<f(a_1,a_2,\cdots,a_k)$. This is because

$$\begin{aligned}\frac{f(a_1,a_2,\cdots,a_k)}{f(a_1,\cdots,a_i+1,\cdots,a_j-1,\cdots,a_k)}&=\sqrt{\frac{a_ia_j^3}{(a_i+1)^3(a_j-1)}}\\&\overset{a_j=a_i+k}{=}\sqrt{\frac{a_i(a_i+k)^3}{(a_i+1)^3(a_i+k-1)}}\\&=\sqrt{\frac{a_i^4+3ka_i^3+3k^2a_i^2+k^3a_i}{a_i^4+(2+k)a_i^3+3ka_i^2+(3k-2)a_i+k-1}}>1,\end{aligned}$$

where the last inequality is due to

$$(2k-2)a_i^3+(3k^2-3k)a_i^2+(k^3-3k+2)a_i\geq 1.$$

We can adjust a_i to a_i+1, a_j to a_j-1 to get a lower value of f. After limited times of such adjustments, $f(a_1,\cdots,a_k)$ will be minimized when the integers $a_1,a_2,\cdots,a_k$ satisfy $\max\{|a_i-a_j|,1\leq i,j\leq k\}\leq 1$.

Similarly, we can easily extend the above analysis to $\frac{1}{a_1}+\cdots+\frac{1}{a_k}$ and prove that it gets minimized if $\max\{|a_i-a_j|,1\leq i,j\leq k\}\leq 1$. □

We next prove our main results Theorem 2

Proof. We first prove that when $t = 1$, the sketching algorithm using $\boldsymbol{S}_{1\times k}$ with one array and k bins (i.e., k counters in the array) achieves $\ln\left(1 + \frac{\beta\alpha^2 k(k-1)}{\sigma^2(n-2)}(1 + \ln(n-k))\right)$-differential privacy with high probability, where β is a constant satisfying $\frac{\alpha^2 k(k-1)}{\sigma^2(n-2)}\left(1 + \ln(n-k)\right) \leq \frac{1}{2} - \frac{1}{\beta}$.

Suppose two input vectors D and D' differ in one element with $D = D_0 + \{c\}$ and $D' = D_0 + \{d\}$. Since the output is independent of the order of compressing each element in D, we consider mapping c (or d) at last. c (or d) is mapped to each bin with equal probability, and they might be flipped by the sign hash functions with a probability $\frac{1}{2}$. Therefore,

$$\begin{aligned}
P(b_1, b_2, \cdots, b_k|D) &= \frac{1}{2k}P(b_1 - c, b_2, \cdots, b_k|D_0) + \frac{1}{2k}P(b_1 + c, b_2, \cdots, b_k|D_0) + \cdots \\
&+ \frac{1}{2k}P(b_1, b_2, \cdots, b_k + c|D_0) + \frac{1}{2k}P(b_1, b_2, \cdots, b_k - c|D_0), \\
P(b_1, b_2, \cdots, b_k|D') &= \frac{1}{2k}P(b_1 - d, b_2, \cdots, b_k|D_0) + \frac{1}{2k}P(b_1 + d, b_2, \cdots, b_k|D_0) + \cdots \\
&+ \frac{1}{2k}P(b_1, b_2, \cdots, b_k + d|D_0) + \frac{1}{2k}P(b_1, b_2, \cdots, b_k - d|D_0).
\end{aligned}$$

Denote $P^n_{(a_1,\cdots,a_k)}$ as the probability that a_i $(1 \leq i \leq k)$ items are mapped to bin i and $\sum_{i=1}^k a_i = n$. Denote $G^n_{(a_1,\cdots,a_k)}$ as the probability density function of a k-dimensional Gaussian $\mathcal{N}\left(0, \begin{bmatrix}\sigma^2 a_1 & & \\ & \ddots & \\ & & \sigma^2 a_k\end{bmatrix}\right)$ and $\sum_{i=1}^k a_i = n$. Note that

$$P(b_1 - c, b_2, \cdots, b_k|D_0) = \sum_{\{a_1,\cdots,a_k\}} P^{n-1}_{(a_1,\cdots,a_k)} G^{n-1}_{(a_1,\cdots,a_k)}(b_1 - c, b_2, \cdots, b_k).$$

We first consider the upper-bound of:

$$\frac{P(b_1 - c, b_2, \cdots, b_k|D_0)}{P(b_1 - d, b_2, \cdots, b_k|D_0)} = \frac{\sum_{\{a_1,\cdots,a_k\}} P^{n-1}_{(a_1,\cdots,a_k)} G^{n-1}_{(a_1,\cdots,a_k)}(b_1 - c, b_2, \cdots, b_k)}{\sum_{\{a_1,\cdots,a_k\}} P^{n-1}_{(a_1,\cdots,a_k)} G^{n-1}_{(a_1,\cdots,a_k)}(b_1 - d, b_2, \cdots, b_k)},$$

where the joint probability distribution of number of items when putting $n-1$ items into k bins $\{b_1, \cdots, b_k\}$ is denoted as $P^{n-1}_{(a_1,\cdots,a_k)} = \frac{(n-1)!}{k^{n-1}\cdot a_1!\cdots a_k!}$. We assume that each bin has at least one item, so there are $\binom{n-2}{k-1}$ ways in total to put $n-1$ items into k bins. Note that probability that each bin has at least one item given k bins and $n-1$ items is $\sum_{j=0}^k (-1)^j \binom{k}{j}\left(1 - \frac{j}{k}\right)^{n-1}$, so the following differential privacy properties with one hash function hold with a high probability at most $\sum_{j=0}^k (-1)^j \binom{k}{j}\left(1 - \frac{j}{k}\right)^{n-1}$.

Denote $\mathbf{x}$ as $\{b_1 - c, b_2, \cdots, b_k\}$, since $e^{-\frac{1}{2}\mathbf{x}^\intercal \Sigma^\intercal \mathbf{x}} \leq 1$, we have:

$$\begin{aligned}
&\frac{\sum_{\{a_1,\cdots,a_k\}} P^{n-1}_{(a_1,\cdots,a_k)} G^{n-1}_{(a_1,\cdots,a_k)}(b_1 - c, b_2, \cdots, b_k)}{\sum_{\{a_1,\cdots,a_k\}} P^{n-1}_{(a_1,\cdots,a_k)} G^{n-1}_{(a_1,\cdots,a_k)}(b_1 - d, b_2, \cdots, b_k)} \qquad (2)\\
\leq &\frac{\sum_{\{a_1,\cdots,a_k\}} \frac{1}{a_1!\cdot a_2!\cdots a_k!} \frac{1}{\sqrt{a_1\cdot a_2\cdots a_k}}}{\sum_{\{a_1,\cdots,a_k\}} \frac{1}{a_1!\cdot a_2!\cdots a_k!} \frac{1}{\sqrt{a_1\cdot a_2\cdots a_k}} e^{\left[-\frac{1}{2\sigma^2}\left(\frac{(b_1-d)^2}{a_1} + \frac{(b_2)^2}{a_2} + \cdots + \frac{(b_k)^2}{a_k}\right)\right]}}\\
\leq &\frac{\sum_{\{a_1,\cdots,a_k\}} \frac{1}{a_1!\cdot a_2!\cdots a_k!} \frac{1}{\sqrt{a_1\cdot a_2\cdots a_k}}}{\sum_{\{a_1,\cdots,a_k\}} \frac{1}{a_1!\cdot a_2!\cdots a_k!} \frac{1}{\sqrt{a_1\cdot a_2\cdots a_k}} e^{\left[-\frac{1}{2\sigma^2}\left(\frac{\gamma}{a_1} + \frac{\gamma}{a_2} + \cdots + \frac{\gamma}{a_k}\right)\right]}}, \qquad (3)
\end{aligned}$$

where $\gamma = \max\{(b_1 - d)^2, b_2^2, \cdots, b_k^2\}$. From the bounded estimation errors of sketching (Lemma 1), we know that $b_i^2 \leq \alpha^2$ $(\forall i)$ with a large probability. So $\gamma \leq 4\alpha^2$ holds with high probability.

Note that (3) can be viewed as an multiplicative inverse of weighted sum on $e^{\left[-\frac{1}{2}\left(\frac{\gamma}{a_1} + \frac{\gamma}{a_2} + \cdots + \frac{\gamma}{a_k}\right)\right]}$ with weights $\frac{1}{a_1!\cdot a_2!\cdots a_k!}\frac{1}{\sqrt{a_1\cdot a_2\cdots a_k}}$. From Lemma 2, we know that both $e^{\left[-\frac{1}{2}\left(\frac{\gamma}{a_1} + \frac{\gamma}{a_2} + \cdots + \frac{\gamma}{a_k}\right)\right]}$ and the weights $\frac{1}{a_1!\cdot a_2!\cdots a_k!}\frac{1}{\sqrt{a_1\cdot a_2\cdots a_k}}$ are maximized simultaneously when $a_1, a_2, \cdots, a_k$ satisfies $\max\{|a_i - a_j|, 1 \leq i, j \leq k\} \leq 1$. If we set all weights to be equal, then the inverse of the weighted sum will become larger, w.h.p. Therefore,

$$\begin{aligned}
\frac{\sum_{\{a_1,\cdots,a_k\}} \frac{1}{a_1!\cdot a_2!\cdots a_k!} \frac{1}{\sqrt{a_1\cdot a_2\cdots a_k}}}{\sum_{\{a_1,\cdots,a_k\}} \frac{1}{a_1!\cdot a_2!\cdots a_k!} \frac{1}{\sqrt{a_1\cdot a_2\cdots a_k}} e^{\left[-\frac{1}{2\sigma^2}\left(\frac{\gamma}{a_1} + \frac{\gamma}{a_2} + \cdots + \frac{\gamma}{a_k}\right)\right]}} &\leq \frac{\binom{n-2}{k-1}}{\sum_{\{a_1,\cdots,a_k\}} e^{\left[-\frac{1}{2\sigma^2}\left(\frac{\gamma}{a_1} + \frac{\gamma}{a_2} + \cdots + \frac{\gamma}{a_k}\right)\right]}}\\
&\leq \frac{\binom{n-2}{k-1}}{\sum_{\{a_1,\cdots,a_k\}} \left(1 - \frac{1}{2\sigma^2}\left(\frac{\gamma}{a_1} + \frac{\gamma}{a_2} + \cdots + \frac{\gamma}{a_k}\right)\right)} \qquad (4)\\
&= \frac{\binom{n-2}{k-1}}{\binom{n-2}{k-1} - \frac{1}{2\sigma^2}\sum_{\{a_1,\cdots,a_k\}}\left(\frac{\gamma}{a_1} + \frac{\gamma}{a_2} + \cdots + \frac{\gamma}{a_k}\right)}\\
&= \frac{1}{1 - \frac{\frac{1}{2\sigma^2}\sum_{\{a_1,\cdots,a_k\}}\left(\frac{\gamma}{a_1} + \frac{\gamma}{a_2} + \cdots + \frac{\gamma}{a_k}\right)}{\binom{n-2}{k-1}}},
\end{aligned}$$

(4) holds because for any real number x, $e^{-x} \geq 1 - x$. (4) also requires $\sum_{\{a_1,\cdots,a_k\}} \left(1 - \frac{1}{2\sigma^2}\left(\frac{\gamma}{a_1} + \frac{\gamma}{a_2} + \cdots + \frac{\gamma}{a_k}\right)\right) \geq 0$, which we will enforce again later.

We next consider to upper-bound $\frac{\frac{\gamma}{2\sigma^2}\sum_{\{a_1,\cdots,a_k\}}\left(\frac{1}{a_1} + \frac{1}{a_2} + \cdots + \frac{1}{a_k}\right)}{\binom{n-2}{k-1}}$.

If we place j items into one specific bin, there are $\binom{n-2-j}{k-2}$ ways to put the remaining $n-1-j$ items to $k-1$ bins. Therefore, in the sum $\sum_{\{a_1,\cdots,a_k\}}\left(\frac{1}{a_1} + \frac{1}{a_2} + \cdots + \frac{1}{a_k}\right)$, for each bin i, $a_i = j$ $(1 \leq j \leq n-k)$ appears

$\binom{n-2-j}{k-2}$ times. And there are k bins, so we have:

$$\frac{\frac{\gamma}{2\sigma^2}\sum_{\{a_1,\cdots,a_k\}}\left(\frac{1}{a_1}+\frac{1}{a_2}+\cdots+\frac{1}{a_k}\right)}{\binom{n-2}{k-1}} = \frac{\frac{\gamma}{2\sigma^2}k\left(1\binom{n-3}{k-2}+\frac{1}{2}\binom{n-4}{k-2}+\cdots+\frac{1}{n-k}\binom{k-2}{k-2}\right)}{\binom{n-2}{k-1}}$$
$$= \frac{\frac{\gamma}{2\sigma^2}k\sum_{i=1}^{n-k}\frac{1}{i}\binom{n-2-i}{k-2}}{\binom{n-2}{k-1}}.$$

By expanding and rearranging the terms, we get:

$$\frac{\frac{\gamma}{2\sigma^2}k\sum_{i=1}^{n-k}\frac{1}{i}\binom{n-2-i}{k-2}}{\binom{n-2}{k-1}} = \frac{\frac{\gamma}{2\sigma^2}k(k-1)}{(n-2)(n-3)\cdots(n-k)}\sum_{i=1}^{n-k}\frac{1}{i}(n-i-2)(n-i-3)\cdots(n-i-(k-1))$$
$$= \frac{\frac{\gamma}{2\sigma^2}k(k-1)}{n-2}\sum_{i=1}^{n-k}\frac{1}{i}\frac{(n-i-2)(n-i-3)\cdots(n-i-(k-1))}{(n-3)(n-4)\cdots(n-k)} \tag{5}$$
$$\leq \frac{\frac{\gamma}{2\sigma^2}k(k-1)}{n-2}\sum_{i=1}^{n-k}\frac{1}{i} \tag{6}$$
$$\leq \frac{\frac{\gamma}{2\sigma^2}k(k-1)}{n-2}(1+\ln(n-k)) \tag{7}$$
$$\leq \frac{2\alpha^2k(k-1)}{\sigma^2(n-2)}(1+\ln(n-k)). \tag{8}$$

(7) holds because $\sum_{i=1}^{k}\frac{1}{i}\leq 1+\ln k$ $(\forall k)$, and (8) is due to $\gamma\leq 4\alpha^2$ (with high probability).

Therefore,

$$\frac{1}{1-\frac{\frac{\gamma}{2\sigma^2}\sum_{\{a_1,\cdots,a_k\}}\left(\frac{1}{a_1}+\frac{1}{a_2}+\cdots+\frac{1}{a_k}\right)}{\binom{n-2}{k-1}}} \leq \frac{1}{1-\frac{2\alpha^2k(k-1)}{\sigma^2(n-2)}(1+\ln(n-k))} \leq 1+\frac{\beta\alpha^2k(k-1)}{\sigma^2(n-2)}(1+\ln(n-k)) \ (\beta>0),$$

where the second inequality holds when $\frac{\alpha^2k(k-1)}{\sigma^2(n-2)}(1+\ln(n-k))\leq\frac{1}{2}-\frac{1}{\beta}$ $(\beta>0)$.

Thus,

$$\frac{P(b_1-c,b_2,\cdots,b_k|D_0)}{P(b_1-d,b_2,\cdots,b_k|D_0)} \leq 1+\frac{\beta\alpha^2k(k-1)}{\sigma^2(n-2)}(1+\ln(n-k)).$$

Similarly, for any i $(1\leq i\leq k)$, we have:

$$\frac{P(b_1,\cdots,b_i-c,\cdots,b_k|D_0)}{P(b_1,\cdots,b_i-d,\cdots,b_k|D_0)} \leq 1+\frac{\beta\alpha^2k(k-1)}{\sigma^2(n-2)}(1+\ln(n-k)),$$

and

$$\frac{P(b_1,\cdots,b_i+c,\cdots,b_k|D_0)}{P(b_1,\cdots,b_i+d,\cdots,b_k|D_0)} \leq 1+\frac{\beta\alpha^2k(k-1)}{\sigma^2(n-2)}(1+\ln(n-k)).$$

Thus,

$$\frac{P(b_1,b_2,\cdots,b_k|D)}{P(b_1,b_2,\cdots,b_k|D')} \leq 1+\frac{\beta\alpha^2k(k-1)}{\sigma^2(n-2)}(1+\ln(n-k)),$$

which indicates that the sketching algorithm $\mathcal{M}$ with input size n using Count Sketch with k bins (k counters) and 1 hash function (1 array) achieves $\ln\left(1+\frac{\beta\alpha^2k(k-1)}{\sigma^2(n-2)}(1+\ln(n-k))\right)$-differential privacy with a large probability, where β is a positive constant satisfying $\frac{\alpha^2k(k-1)}{\sigma^2(n-2)}(1+\ln(n-k)) \leq \frac{1}{2}-\frac{1}{\beta}$. Since the t counter arrays are pairwise independent, it follows that Count Sketch with k bins (k counters) and t arrays achieves $t\cdot\ln\left(1+\frac{\beta\alpha^2k(k-1)}{\sigma^2(n-2)}(1+\ln(n-k))\right)$-differential privacy with a large probability, where β is a positive constant satisfying $\frac{\alpha^2k(k-1)}{\sigma^2(n-2)}(1+\ln(n-k)) \leq \frac{1}{2}-\frac{1}{\beta}$. □

B Proof for Theorem 3

Theorem: (Convergence of `DiffSketch` in distributed SGD) Assume that $E\left[\|g_k\|^2\right] \leq G^2$ for any input stochastic gradient g_k on worker k with dimension n, $E\left[\|x_i - x^*\|^2\right] \leq D^2$ at any iteration i, and the local model at device k $f_k(x)$ is convex. Choose the step-size at round i $\eta_i = \frac{c}{\sqrt{i}}$ where c is a pre-defined positive number. Using a Count Sketch with t hash functions and k bins, with probability $1-\delta$, we have the following convergence rate on the global objective F:

$$F(\bar{x}_i) - F(x^*) \leq \frac{\frac{D^2}{2c} + c\sqrt{\frac{i+1}{i}}\left(n\mu^2+1\right)G^2}{\sqrt{i}},$$

where x^* is the optimal solution to (1), $k = O\left(\frac{e}{\mu^2}\right)$, $t = O\left(\ln(\frac{1}{\delta})\right)$, and $\bar{x}_i = \frac{1}{i}\sum_{j=1}^{i} x_j$.

Proof. Because the local loss function f_k and the global loss function F are convex, we have

$$\langle \nabla f_k(x_i), x_i - x^* \rangle \geq f_k(x_i) - f_k(x^*),$$

and

$$\langle \nabla F(x_i), x_i - x^* \rangle \geq F(x_i) - F(x^*).$$

We use $\tilde{g}_k$ to denote the estimated stochastic gradient on worker k after querying the sketch table at the current iteration i (we omit the subscript i for cleaner notations). Suppose K worker are selected at each updating round, the updating rule is $x_{i+1} = x_i - \eta_i\frac{1}{K}\sum_{k=1}^{K}\tilde{g}_k$, and we have

$$\begin{aligned}
\mathbb{E}\left[\|x_{i+1}-x^*\|^2\right] &= \mathbb{E}\left[\left\|x_i - \eta_i\frac{1}{K}\sum_{k=1}^{K}\tilde{g}_k - x^*\right\|^2\right] \\
&= \mathbb{E}\left[\|x_i - x^*\|^2\right] - 2\eta_i\mathbb{E}\left[\left\langle \frac{1}{K}\sum_{k=1}^{K}\tilde{g}_k, x_i - x^*\right\rangle\right] + \eta_i^2\mathbb{E}\left[\left\|\frac{1}{K}\sum_{k=1}^{K}\tilde{g}_k\right\|^2\right].
\end{aligned}$$

From Lemma 1, we know that the estimation error of Count Sketch is bounded. In particular, it holds that with probability $p \geq 1-\delta$,

$$\mathbb{E}\left[\|\tilde{g}_k - g_k\|^2\right] \leq \mathbb{E}\left[n\left(\mu\|g_k\|_2\right)^2\right] \leq n\mu^2G^2,$$

where $k = O\left(\frac{e}{\mu^2}\right)$ and $t = O\left(\ln\left(\frac{1}{\delta}\right)\right)$. Thus,

$$\begin{aligned}
\mathbb{E}\left[\|\tilde{g}_k\|^2\right] = \mathbb{E}\left[\|\tilde{g}_k - g_k + g_k\|^2\right] &\leq \mathbb{E}\left[2\left(\|\tilde{g}_k - g_k\|^2 + \|g_k\|^2\right)\right] \\
&\leq 2\left(n\mu^2+1\right)G^2.
\end{aligned}$$

Further, we have

$$\begin{aligned}\mathbb{E}\left[\left\|\frac{1}{K}\sum_{k=1}^{K}\tilde{g}_k\right\|^2\right] &= \frac{1}{K^2}\mathbb{E}\left[\left\|\sum_{k=1}^{K}\tilde{g}_k\right\|^2\right] \\ &\leq \frac{1}{K^2}\cdot K\cdot\sum_{k=1}^{K}\mathbb{E}\left[\|\tilde{g}_k\|^2\right] \\ &\leq 2\left(n\mu^2+1\right)G^2.\end{aligned}$$

In addition, as sketches produce an unbiased estimation of the stochastic gradient, which is an unbiased estimation of the true gradient, we have $\mathbb{E}\left[\tilde{g}_k\right]=\mathbb{E}\left[\nabla f_k\right]$ (the expectation is with respect to the randomly sampled data point and the randomized sketch algorithm). And $\mathbb{E}\left[\frac{1}{K}\sum_{k=1}^{K}\tilde{g}_k\right]=\mathbb{E}\left[\frac{1}{K}\sum_{k=1}^{K}\nabla f_k\right]=\mathbb{E}\left[\nabla F(x_i)\right]$ Applying the bounded variance and the unbiased compression, it follows:

$$\begin{aligned}\mathbb{E}\left[\|x_{i+1}-x^*\|^2\right] &\leq \mathbb{E}\left[\|x_i-x^*\|^2\right]-2\eta_i\mathbb{E}\left[\langle\nabla F(x_i),x_i-x^*\rangle\right]+2\eta_i^2\left(n\mu^2+1\right)G^2 \\ &\leq \mathbb{E}\left[\|x_i-x^*\|^2\right]-2\eta_i\left(F(x_i)-F(x^*)\right)+2\eta_i^2\left(n\mu^2+1\right)G^2 \\ \Rightarrow 2\left(F(x_i)-F(x^*)\right) &\leq \frac{1}{\eta_i}\mathbb{E}\left[\|x_i-x^*\|^2\right]-\frac{1}{\eta_i}\mathbb{E}\left[\|x_{i+1}-x^*\|^2\right]+2\eta_i\left(n\mu^2+1\right)G^2.\end{aligned}$$

Summarizing the inequalities for $j=1,\cdots,i$, we get

$$\begin{aligned}2\sum_{j=i}^{i}\left(F(x_j)-F(x^*)\right) &\leq \frac{1}{\eta_j}\mathbb{E}\left[\|x_1-x^*\|^2\right]+\sum_{j=2}^{i}\left(\frac{1}{\eta_j}-\frac{1}{\eta_{j-1}}\right)\mathbb{E}\left[\|x_j-x^*\|^2\right]+\sum_{j=1}^{k}2\eta_j\left(n\mu^2+1\right)G^2 \\ &\leq \frac{1}{\eta_j}D^2+\sum_{j=2}^{i}\left(\frac{1}{\eta_j}-\frac{1}{\eta_{j-1}}\right)D^2+\sum_{j=1}^{i}2\eta_j\left(n\mu^2+1\right)G^2 \\ &\leq \frac{\sqrt{i}}{c}D^2+2c\sqrt{i+1}\left(n\mu^2+1\right)G^2.\end{aligned}$$

From Jensen's inequality, we have

$$\sum_{j=1}^{i}F(x_j)-F(x^*)\geq F(\bar{x}_i)-F(x^*),$$

which indicates that

$$F(\bar{x}_i)-F(x^*)\leq\frac{\frac{\sqrt{i}}{2c}D^2+c\sqrt{i+1}\left(n\mu^2+1\right)G^2}{i}.$$

This completes the proof. □

C Evaluation Details

C.1 Implementation Details

We first describe the implementation details for each experiment comparing with different baselines.

Figure 3. We perform error correction for both `DiffSketch` and the baseline method of adding Laplacian noise after sketching. At each round, each local worker computes the gap between the latest local gradient and the (estimated) aggregated gradients. Since data are identically distributed across all workers, if the estimation is correct, we expect that the gap should be small. Therefore, each worker sets half of the queried results to zero if he sees a large gap. For `DiffSketch`, we also generate a noise vector drawn from the same Gaussian as the raw gradients, and append that noise vector to the input for compression. The noise vector increases the size of the input, thus boosting the privacy for sketches.

Figure 4. We do not perform error correction for any methods, since the local gradients would be stale in federated learning due to device sampling. Similarly, we append a noise vector after the raw model updates.

Figure 2. We use the δ, s, p values suggested in Agarwal et al. [3]. We set up 6,000 workers (MNIST*) for this experiment, and select 10 at each round. We use a different variant of MNIST because the privacy bound of `cpSGD` will be tighter with a larger number of total workers. For `cpSGD`, we calculate the k, m values (corresponding to the k, m parameters in the original paper) based on the given ε and compression ratios, as summarized in the following table 1.

Table 1: Parameter configuratioins for `cpSGD`

ε	Bits (Compression)	k	m
5	4 (10×)	12	4
5	5 (6×)	23	9
10	4 (10×)	14	2
10	5 (6×)	28	4

Datasets. We summarize the statistics of the datasets below.

Table 2: Statistics of Datasets

Dataset	**Workers**	**Param.**	**Samples/device**	
			mean	stdev
MNIST	10	7,850	200	0
Shakespeare	46	39,720	742	548
MNIST*	6,000	7,850	10	0

Hyper-parameters. We assume that the input gradients are bounded by a constant α with a 90% probability; therefore, we dynamically choose the 90th percentile value of the local gradient vector as α on each worker. For all experiments, we use a batch size of 10. The learning rates of MNIST, MNSIT*, and Shakespeare are 0.01, 0.01, and 0.8. For each comparison, we fix the mini-batch orders and (if needed) the selected devices per round.

C.2 Real Gradient Distributions

We plot the real gradient distributions for MNIST and model updates distribution for Shakespeare without compression in Figure 7 and Figure 8. When applying `DiffSketch`, we observe that their gradient/model update distributions throughout the training across are similar. In MNIST, there are ten workers participating in training at each round and the data on the ten workers are identically distributed. We randomly select one worker at each round and report the gradient distribution from that worker. We repeat this for ten rounds. In Shakespeare, we only sample one device per round; therefore we show the model update distributions of ten selected devices in ten rounds.

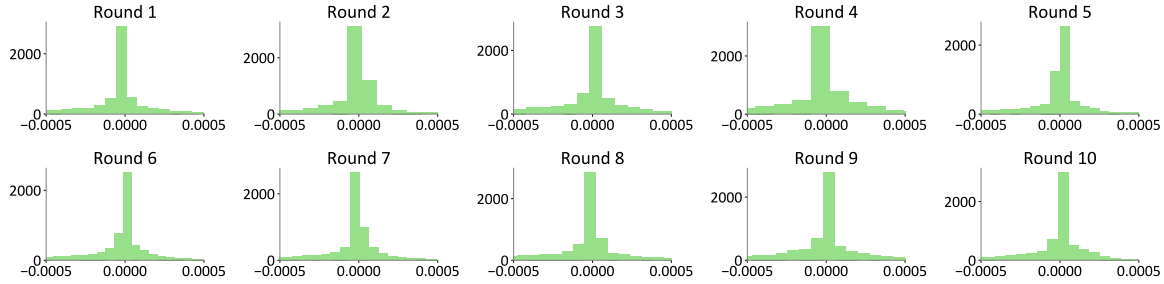

Figure 7: Gradient distributions of MNIST

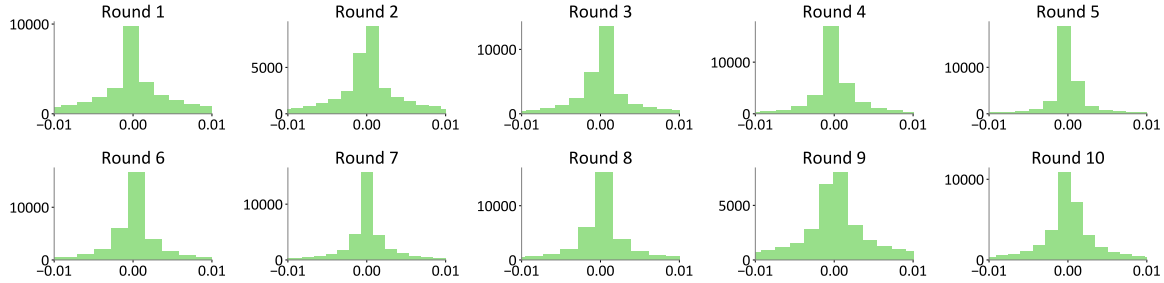

Figure 8: Distributions of model updates of Shakespeare

C.3 Privacy Parameter (ε) Distributions Across All Rounds

As mentioned before, due to the inherent privacy properties of sketches, a small amount of (Laplacian) noise is sufficient to provide a certain level of differential privacy guarantees. In Figure 9, we visualize the histogram of the local ε values on the MNIST dataset (with a 75× compression ratio and $\varepsilon = 1$) across all rounds. We see that most of the ε values guaranteed by sketches are bounded by the input privacy requirement 0.1, and we need to add additional Laplacian noise in a few cases to obtain a consistent privacy bound.

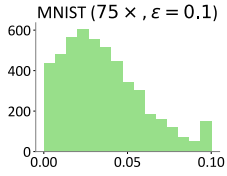

Figure 9: Local ε values across all rounds when the input ε requirement is 0.1.