Skellam Mixture Mechanism: a Novel Approach to Federated Learning with Differential Privacy

Ergute Bao
National University of Singapore
bao@u.nus.edu

Yizheng Zhu
National University of Singapore
yzhu@nus.edu.sg

Xiaokui Xiao
National University of Singapore
xkxiao@nus.edu.sg

Yin Yang
Hamad Bin Khalifa University
yyang@hbku.edu.qa

Beng Chin Ooi
National University of Singapore
ooibc@comp.nus.edu.sg

Benjamin Hong Meng Tan
A*STAR, Singapore
benjamin_tan@i2r.a-star.edu.sg

Khin Mi Mi Aung
A*STAR, Singapore
mmaung@i2r.a-star.edu.sg

ABSTRACT

Deep neural networks have strong capabilities of memorizing the underlying training data, which can be a serious privacy concern. An effective solution to this problem is to train models with *differential privacy* (*DP*), which provides rigorous privacy guarantees by injecting random noise to the gradients. This paper focuses on the scenario where sensitive data are distributed among multiple participants, who jointly train a model through *federated learning*, using both *secure multiparty computation* (*MPC*) to ensure the confidentiality of each gradient update, and differential privacy to avoid data leakage in the resulting model. A major challenge in this setting is that common mechanisms for enforcing DP in deep learning, which inject *real-valued noise*, are fundamentally incompatible with MPC, which exchanges *finite-field integers* among the participants. Consequently, most existing DP mechanisms require rather high noise levels, leading to poor model utility.

Motivated by this, we propose *Skellam mixture mechanism* (SMM), a novel approach to enforcing DP on models built via federated learning. Compared to existing methods, SMM eliminates the assumption that the input gradients must be integer-valued, and, thus, reduces the amount of noise injected to preserve DP. The theoretical analysis of SMM is highly non-trivial, especially considering (i) the complicated math of DP deep learning in general and (ii) the fact that the mixture of two Skellam distributions is rather complex. Extensive experiments on various practical settings demonstrate that SMM consistently and significantly outperforms existing solutions in terms of the utility of the resulting model.

PVLDB Reference Format:
Ergute Bao, Yizheng Zhu, Xiaokui Xiao, Yin Yang, Beng Chin Ooi, Benjamin Hong Meng Tan, and Khin Mi Mi Aung. Skellam Mixture Mechanism: a Novel Approach to Federated Learning with Differential Privacy. PVLDB, 15(11): 2348-2360, 2022.
doi:10.14778/3551793.3551798

Proceedings of the VLDB Endowment, Vol. 15, No. 11 ISSN 2150-8097.
doi:10.14778/3551793.3551798

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/SkellamMixtureMechanism/SMM.

1 INTRODUCTION

Deep neural networks, especially large-scale ones such as GPT-3 [10], are known for their excellent memorization capabilities [22, 43, 52]. However, it is rather difficult to control what exactly the neural net memorizes, and unintended data memorization can be a serious concern when the underlying training data contains sensitive information [12]. For instance, consider a bank that trains a GPT-like language model on call center transcripts. Due to data memorization, it is possible to extract sensitive information by letting the model auto-complete a prefix, e.g., "my account number is: __". Clearly, if such a model (or its API) is exposed to the adversary, it becomes a ligation machine as attackers can attempt with various prefixes to extract sensitive data, and subsequently sue the bank for privacy violations. Shokri et al. [42] report that simple and intuitive measures often fail to provide sufficient protection, and the only way found to completely address the issue is to train the model with the rigorous guarantees of *differential privacy* (*DP*) [18].

This paper focuses on the scenario that multiple individual participants jointly train a machine learning model using *federated learning* (*FL*) [34] through distributed stochastic gradient descent (SGD) [1, 15, 17, 33]. Specifically, in every iteration, each individual computes the gradients with respect to the current model weights based on her own data; then, gradients from all participants are aggregated to update the model. Note that the gradients from each individual may reveal sensitive information about her private dataset [35, 38, 41, 42, 51]. A common approach to addressing this problem is by employing a *secure multiparty computation* (*MPC*) protocol [5, 7, 13, 16, 23, 27, 50], which computes the aggregate gradients while preserving the confidentiality of the gradients from each individual participant. One advantage of MPC is that it does not require a trusted third party, which can be difficult to establish in some applications, e.g., in finance and healthcare.

Note that although MPC protects individuals' privacy in the gradient update process by concealing the gradient values of each participant, it does not provide any protection against data extraction attacks caused by unintended data memorization [20, 35, 44, 45].

As mentioned earlier, an effective methodology to defend against such attacks is to perturb the gradients to satisfy differential privacy [42]. Since there is no trusted third-party in our setting, such gradient perturbations need to be done in a decentralized fashion. In particular, each FL participant first adds noise to her own gradients; then, the participants collectively aggregate their noisy gradients, through a cryptographically secure protocol, e.g., SecAgg [9], which ensures that (i) the server, who later updates the model based on the aggregated outcome, learns nothing about the perturbed gradients but the outcome itself, and (ii) no participant learns private information about other participants' data, except for the aggregated outcome. Hence, the privacy cost incurred at each iteration only depends upon the sensitivity of the sum of the gradients, as well as the distribution of the aggregated noise. This framework is referred to as distributed differential privacy [24, 28], elaborated in Section 2.4.

Although gradient perturbation under DP has been studied in previous work, it is far from trivial to adapt centralized DP solutions to the distributed setting. For instance, consider the classic DPSGD algorithm [2], in which a centralized party injects random Gaussian noise to the gradient sum in each iteration of the model training process. There are two major challenges for adapting DPSGD to distributed DP. First, the Gaussian distribution is defined over the domain of all real numbers, whereas existing MPC protocols, to our knowledge, require inputs to be represented as *integers* (more precisely, finite field elements) [8, 9, 39]. Although we could sample a real value from the Gaussian distribution, and then quantize the value to an integer, the resulting quantized samples would no longer follow the Gaussian distribution, which, strictly speaking, invalidates the proof in [2] that the method satisfies DP. Second, even if we ignore the privacy risk of using quantized Gaussian samples in DPSGD [2], the privacy analysis of the algorithm also relies on certain mathematical properties of the Gaussian distribution, which no longer hold with quantized samples. Such properties include: (i) the sum of n values sampled from i.i.d. unit-variance Gaussian distribution follows the Gaussian distribution with variance n, and (ii) there exists a tight upper bound for the Rényi divergence [36] between two Gaussian distributions. These issues have been largely neglected by earlier distributed DP solutions, e.g., [24, 46, 47].

Recently, three methods [3, 4, 28] were proposed to address the above problem. A common high-level idea of these methods is to require that each participant of FL inject symmetric integer-valued noise (e.g., binomial noise in [4]) to the gradients during each iteration of the training process. Here, the original values of the gradients are assumed to have bounded norms and are integer-valued, elaborated soon. Next, we point out the main drawback of existing methods, which motivates our proposed solution. Recall the two assumption made above: gradients have bounded norms and are integer-valued. While the bounded norm assumption can be enforced by gradient clipping as is done in DPSGD [2], the integer-valued gradients assumption requires a more complicated pre-processing step accompanied by a careful privacy analysis. Here, we briefly explain how existing works enforce this assumption, and we defer a detailed discussion to Section 5. Ref. [4] stochastically rounds the gradients to integers. For example, if $x = \{0.01, 0.01, \ldots, 0.01\} \in \mathbb{R}^d$, then each dimension of x is rounded to 1 with 0.01 probability, and to 0 with 0.99 probability. While the rounded gradient's expectation equals the original one, the norm of the rounded gradient could be significantly larger than the original one. In our example, $x = \{0.01, 0.01, \ldots, 0.01\} \in \mathbb{R}^d$ could be rounded to $\{1, 1, \ldots, 1\}$, causing an almost $\sqrt{d}$ the increase in $\mathcal{L}_2$ norm. Such an increased sensitivity leads to higher amount of perturbations required to satisfy DP, which, in turn, leads to reduced model utility. Ref. [28] proposes a more complicated conditional rounding process to alleviates (to a limited degree) the problem of increased sensitivity, at the expense of introducing additional bias terms to the resulting gradients, as explained in Section 5.

Our Contributions. In this work, we propose the *Skellam mixture mechanism* (SMM), a new solution for enforcing distributed differential privacy for federated learning. SMM works by injecting random noise drawn from the mixture of two shifted symmetric Skellam distributions. *Unlike existing solutions,* SMM *does not require its inputs (i.e., the gradients in FL) to be integer-valued.* This eliminates the need for the process of stochastic rounding the gradients, leading to lower noise level required to satisfy DP, and, thus, higher utility of the resulting model. In particular, with carefully selected mixture coefficients and Skellam distribution parameters, SMM produces private and unbiased integer-valued gradient aggregates for updating the model. We prove that SMM satisfies both Rényi-DP and (ϵ, δ)-DP, defined in Section 2. Meanwhile, SMM is compatible with the DPSGD [2] framework and its moment accountant analysis technique, leading to tight bounds on the privacy loss analysis, similar to our competitors [3, 28].

The privacy analysis of SMM is rather challenging, and is a major contribution of the paper. Note that although the Skellam distribution has been used in previous solution [3], the privacy analysis in [3] does not apply to our setting, since the random noise in SMM is sampled from a mixture of two shifted symmetric Skellam distributions, which is more complex than the single Skellam distribution as in [3]. Further, it is unclear how to derive a tight privacy bound for SMM using the results and mathematical tools built in [3]. One major reason is the privacy bound in Ref. [3] for the Skellam noise requires upper-bounding both the $\mathcal{L}_1$ and $\mathcal{L}_2$ sensitivity of the input, and it is unclear how this bound can be extended to the case of a Skellam mixture distribution. To tackle this problem, we first derive a cleaner privacy bound for the Skellam noise that only involves the $\mathcal{L}_2$ sensitivity, which is the foundation of our privacy analysis for SMM. Our analysis technique for the Skellam distribution is of independent interest, and can be applied to the setting of [3] to improve its privacy bounds by removing the dependency on the input's $\mathcal{L}_1$ sensitivity.

We apply SMM to federated learning with distributed SGD, and present the complete training algorithm. As mentioned above, SMM improves model utility by eliminating the step of rounding the gradients, which often significantly increases the sensitivity of the inputs, especially for large models. Extensive experiments using benchmark datasets demonstrate that SMM achieves consistent and significant utility gains over its competitors, under a variety of settings with different privacy and communication constraints.

2 PRELIMINARIES

2.1 Skellam Distribution

A random variable Y follows a Poisson distribution of parameter λ if its probability distribution is $\Pr[Y = k] = \frac{\exp(-\lambda)\lambda^k}{k!}, k = 0, 1, 2, \ldots$. Both the mean and variance of Y is λ. A random variable Z follows a Skellam distribution if it is the difference between two independent Poisson variables Y_1 and Y_2. In this work, we restrict our attention to the case where Y_1 and Y_2 have the same parameter λ. In this case, the probability distribution of Z is

$$\Pr[Z = k] = \exp(-2\lambda) I_{|k|}(2\lambda), k = 0, \pm 1, \pm 2, \ldots,$$

where $I_v(u) \triangleq \sum_{h=0}^{\infty} \frac{1}{h!\Gamma(h+v+1)} \left(\frac{u}{2}\right)^{2h+v}$ is the modified Bessel function of the first kind. We write that $Z \sim \mathrm{Sk}(\lambda, \lambda)$. By linearity of expectation, Z has mean 0 and variance 2λ.

Skellam distributions have an important property: they are "additive", in the sense that for any two independent Skellam random variables $Z_1 \sim Sk(\lambda_1, \lambda_1)$, and $Z_2 \sim Sk(\lambda_2, \lambda_2)$, their sum $Z_1 + Z_2$ follows a Skellam distribution $Sk(\lambda_1 + \lambda_2, \lambda_1 + \lambda_2)$. This property is crucial in our analysis of the privacy guarantee of the Skellam mixture noise used in our solution.

2.2 Rényi Divergence

Definition 1 (Rényi Divergence [48]). *Assuming that distributions P and Q are defined over the same domain, and P is absolute continuous with respect to Q, then the Rényi divergence of P from Q of finite order $\alpha \in (0, 1) \cup (1, \infty)$ is defined as:*

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

where we adopt the convention that $\frac{0}{0} = 0$ and $\frac{y}{0} = \infty$ for any $y > 0$, and the logarithm is with base e.

We next present some useful properties of Rényi divergence.

Theorem 1 (Convexity [48]). *For any order $\alpha \in [0.\infty]$ and $0 < \lambda < 1$, Rényi divergence is convex in its second argument. That is, for any probability distributions P, Q_0, and Q_1*

$$D_\alpha(P \,\|\, (1-\lambda) \cdot Q_0 + \lambda \cdot Q_1) \leq (1-\lambda) \cdot D_\alpha(P \,\|\, Q_0) + \lambda \cdot D_\alpha(P \,\|\, Q_1).$$

Theorem 2 (Joint Quasi-convexity [48]). *For any order $\alpha \in [0.\infty]$ and $0 < \lambda < 1$, Rényi divergence is jointly quasi-convex in its arguments, i.e., for any two pairs of probability distributions (P_0, Q_0), and (P_1, Q_1)*

$$D_\alpha((1-\lambda) \cdot P_0 + \lambda \cdot P_1 \,\|\, (1-\lambda) \cdot Q_0 + \lambda \cdot Q_1) \leq \max\{D_\alpha(P_0 \,\|\, Q_0), D_\alpha(P_1 \,\|\, Q_1)\}.$$

2.3 Differential Privacy

We say that two datasets X and X' are neighboring if one can be obtained by adding or removing one tuple from the other. The main idea of differential privacy (DP) is to ensure that the outcomes of a randomized mechanism on neighboring datasets are always similar; intuitively, this provides plausible deniability on whether a given data record x belongs to the dataset X or not, and, thus, protects the privacy of the individual whose record is x. A classic definition of differential privacy is (ϵ, δ)-DP [18], as follows.

Definition 2 ((ϵ, δ)-Differential Privacy [18]). *A randomized mechanism $\mathcal{M}$ satisfies (ϵ, δ)-differential privacy (DP) if*

$$\Pr[\mathcal{M}(X) \in O] \leq \exp(\epsilon) \cdot \Pr[\mathcal{M}(X') \in O] + \delta,$$

for any set of output $O \subseteq Range(\mathcal{M})$ and any neighboring datasets X and X'.

Note that (ϵ, δ)-DP can be considered as a worst-case privacy guarantee for a mechanism, as it enforces an upper bound on the probability ratio of all possible outcomes. An alternative definition is Rényi differential privacy (RDP) [36], which is built upon the concept of Rényi divergence, considers the average case privacy guarantee instead.

Definition 3 (Rényi Differential Privacy [36]). *A randomized mechanism $\mathcal{M}$ satisfies (α, τ)-Rényi differential privacy (RDP) if $D_\alpha(\mathcal{M}(X) \,\|\, \mathcal{M}(X')) \leq \tau$ for all neighboring datasets X and X'.*

Given a function of interest, the canonical way to make it differentially private is to perturb its outcome through noise injection. Specifically, the scale of the noise should be calibrated to the sensitivity of the function of interest [18], formally defined as follows.

Definition 4 (Sensitivity). *The sensitivity $S(F)$ of a function $F : \mathcal{D} \to \mathbb{R}^d$, denoted as $S(F)$, is defined as*

$$S(F) = \max_{X \sim X'} \|F(X) - F(X')\|,$$

where $X \sim X'$ denotes that X and X' are neighboring datasets, and $\|\cdot\|$ is a norm.

In particular, injecting continuous Gaussian noise sampled from $\mathcal{N}(0, \sigma^2)$ to each dimension of function F satisfies $(\alpha, \frac{\alpha S^2(F)}{2\sigma^2})$-RDP [36], where $S(F)$ stands for the $\mathcal{L}_2$ sensitivity of function F. In many applications (e.g., training neural networks with SGD), we also need to analyze the overall privacy guarantee of a mechanism consisting of multiple components. We have the following composition and sub-sampling lemmata for RDP mechanisms.

Lemma 1 (Composition of RDP Mechanisms [36]). *If mechanisms $\mathcal{M}_1, \ldots, \mathcal{M}_T$ satisfy $(\alpha, \tau_1), \ldots, (\alpha, \tau_T)$-RDP, respectively, then, $\mathcal{M}_1 \circ \ldots \circ \mathcal{M}_T$ satisfies $(\alpha, \sum_{t=1}^{T} \tau_i)$-RDP.*

Lemma 2 (Subsampling for RDP [37, 53]). *Let $\mathcal{M}$ be a mechanism that satisfies $(l, \tau(l))$-RDP for $l = 2, \ldots, \alpha$ ($\alpha \in \mathbb{Z}, \alpha \leq 2$), and S_q be a procedure that uniformly samples each record of the input data with probability q. Then $\mathcal{M} \circ S_q$ satisfies (α, τ)-RDP with*

$$\tau = \frac{1}{\alpha - 1} \cdot \log \left((1-q)^{\alpha-1}(\alpha q - q + 1) + \sum_{l=2}^{\alpha} \binom{\alpha}{l} (1-q)^{\alpha-l} q^l e^{(l-1)\tau(l)} \right).$$

Finally, any mechanism that satisfies (α, τ)-RDP also satisfies (ϵ, δ)-DP, for values of ϵ and δ as follows.

Lemma 3 (Converting (α, τ)-RDP to (ϵ, δ)-DP [11]). *For any $\alpha \in (1, \infty)$, if $D_\alpha(\mathcal{M}(X) \,\|\, \mathcal{M}(X')) \leq \tau$ for any neighboring databases X and X', then $\mathcal{M}$ satisfies (ϵ, δ)-DP for*

$$\epsilon = \tau + \frac{\log(1/\delta) + (\alpha - 1)\log(1 - 1/\alpha) - \log(\alpha)}{\alpha - 1}.$$

2.4 Distributed Differential Privacy

The original, centralized differential privacy framework [18] assumes a trusted data curator, who stores the entire private dataset and injects random noise in its response to a query, e.g., the sum query, which computes $\sum_{i=1}^{n} x_i$ given input dataset $X = (x_1, \ldots, x_n)$. The released outcome satisfies (centralized) DP, when the scale of the noise injected is calibrated (by the centralized data curator) to the sensitivity of the query. In this work, we focus on the distributed differential privacy framework [14, 21, 24, 28], which involves multiple participants. Each participant injects a random noise to her own data or query response. After that, all participants collectively run an MPC protocol to amplify the privacy guarantee by hiding the identities of the participants. We follow the same threat model as in previous work [28]. In particular, all participants are honest (i.e., they strictly follow the protocol) but curious (i.e., each participant tries to learn private information from another participant), and it is assumed that no two participants collude. In this paper, we focus on SecAgg [9] as the MPC protocol, which aggregates inputs from participants in a crpytographically secure manner under our threat model. Specifically, SecAgg ensures that no one (including the participants) can infer any information about the private inputs other than its released output. The output of SecAgg should satisfy DP, such that it can be distributed the same way as the result from a centralized DP solution. Accordingly, the scale of the overall noise injected to the data or query result needs to be calibrated to the sensitivity of each participant's input. In other words, distributed DP obtains the same privacy-utility trade-off as in centralized DP setting, without relying on a trusted third party.

3 SKELLAM MIXTURE MECHANISM

Section 3.1 formalizes the problem of distributed sum estimation under distributed DP, and Section 3.2 presents the proposed *Skellam Mixture Mechanism* (SMM) for this problem. Section 3.3 presents the foundation of the privacy guarantee of SMM. Section 3.4 establish the privacy and utility guarantees of SMM. Then, in Section 4, we apply SMM to our main problem setting: differentially private federated learning.

3.1 Distributed Sum Estimation with Privacy

Suppose that a multi-dimensional dataset $X = (x_1, \ldots, x_n)$ is distributed to n individuals (referred to as *participants* in the following), where participant i possesses data point $x_i \in \mathbb{R}^d$, for $i = 1, \ldots, n$. An un-trusted *server* aims to compute the (approximate) sum of the dataset, i.e., $\bar{x} = \sum_{i=1}^{n} x_i$, from the participants. Agarwal et al. [4] propose a general framework for solving the distributed sum estimation problem with distributed DP. In this framework, each participant first perturbs her data x_i with noise Z_i: $x_i^* \leftarrow x_i + Z_i$. Next, a secure aggregation protocol SecAgg [9], run as a black box by the participants, sums up the noisy values x_i^* from all participants, and outputs to the server the result $\bar{x}^* \leftarrow SecAgg(x_1^*, \ldots, x_n^*)$.

According to Ref. [9], SecAgg ensures that no participant (or the server) learns any information about another participant's private data. Hence, it suffices to derive the privacy and utility guarantees of the following mechanism $\mathcal{M}$, which injects n independent random noises Z_i to the exact sum:

$$\mathcal{M}(x_1, \ldots, x_n) := \sum_{i=1}^{n} x_i + \sum_{i=1}^{n} Z_i.$$

In terms of privacy, we focus on the RDP definition (Definition 3), which can be converted to the classic (ϵ, δ)-DP (Definition 2) through Lemma 3. In particular, we want that for all neighboring datasets X, X',

$$D_\alpha(\mathcal{M}(X) \,\|\, \mathcal{M}(X')) \le \tau,$$

for some $\alpha > 1$. We measure the error of $\mathcal{M}$ by

$$Err_{\mathcal{M}} = \max_{X \subset \mathbb{R}^d} \frac{1}{d} \mathbb{E} \left\| \mathcal{M}(X) - \sum_{x \in X} x \right\|_2^2,$$

where the expectation is taken over the randomness in $\mathcal{M}$.

Algorithm 1: One-dimensional Skellam mixture mechanism (1SMM)

Input: A set of private values $\{x_1, \ldots, x_n \mid x_i \in \mathbb{R}\}$.
Parameters: Noise parameter λ.
1 **for** $i \in 1..n$ **do**
2 $p_i = x_i - \lfloor x_i \rfloor$.
3 Sample y_i from a Bernoulli trial with success probability p_i
4 **if** $y_i = 0$ **then**
5 $x_i^* \leftarrow \lfloor x_i \rfloor + Sk(\lambda, \lambda)$.
6 **else**
7 $x_i^* \leftarrow \lfloor x_i \rfloor + 1 + Sk(\lambda, \lambda)$.
8 $\bar{x}^* \leftarrow SecAgg((x_1^*, \ldots, x_n^*))$.
Output: $\bar{x}^*$.

3.2 Skellam Mixture Noise

We first consider the case when each participant's data point x_i is one-dimensional. Algorithm 1 shows the pseudo-code of our *one-dimensional Skellam mixture mechanism (*1SMM*)* for this case. Each participant i first independently flips a coin with heads probability $p_i := x_i - \lfloor x_i \rfloor$ (Lines 2 and 3). If it is tails, then the participant perturbs $\lfloor x_i \rfloor$ with a noise following the Skellam distribution $Sk(\lambda, \lambda)$ (Lines 4 and 5); otherwise, the participant perturbs $\lfloor x_i \rfloor + 1$ (i.e., $\lceil x_i \rceil$) with a noise following the Skellam distribution $Sk(\lambda, \lambda)$ (Lines 6 and 7). Note that, by the definitions of x_i^* and the Skellam distribution, x_i^* is guaranteed to be an integer. Finally, SecAgg aggregates the noisy values from all the participants (Line 9), and the estimated sum $\bar{x}^*$ is released to the server. For the case in which each participant's data point x_i is multidimensional, we simply invoke Algorithm 1 for each dimension independently to obtain a noisy sum of that dimension, as outlined in Algorithm 2.

The result of Algorithm 1 may appear rather difficult to analyze at first sight, as there are 2^n possible outcomes of the Bernoulli trials by all participants. An import insight in our analysis is that to derive the utility guarantee of Algorithm 1, it suffices to consider each participant independently. First, note that the perturbed value x_i^* follows a mixture of two shifted symmetric Skellam distributions whose shifted mean values equal $\lfloor x_i \rfloor$ and $\lceil x_i \rceil$, respectively, and the variance of each distribution equals 2λ. In addition, observe that the weights associated with the mixture distributions are $1 - x_i + \lfloor x_i \rfloor$ and $x_i - \lfloor x_i \rfloor$, respectively. Consequently, the expectation of x_i^* equals

Algorithm 2: Multi-dimensional Skellam mixture mechanism (dSMM)

Input: A set of private values $\{x_1, \ldots, x_n \mid x_i \in \mathbb{R}^d\}$.
Parameters: Noise parameter λ, data dimension d.

```
for i ∈ 1..n do
    for j ∈ 1..d do
        p_{i,j} = x_{i,j} − ⌊x_{i,j}⌋.
        Sample y_{i,j} from a Bernoulli trial with success probability p_{i,j}.
        if y_{i,j} = 0 then
            x*_{i,j} ← ⌊x_{i,j}⌋ + Sk(λ, λ).
        else
            x*_{i,j} ← ⌊x_{i,j}⌋ + 1 + Sk(λ, λ).
x̄* ← SecAgg((x*_1, ..., x*_n)).
```

Output: $\bar{x}^*$.

x_i. A corner case is that x_i is an integer. In this case, the perturbed x_i^* can be seen as injecting symmetric Skellam noise $Sk(\lambda, \lambda)$ to x_i itself only. By the linearity of expectation, the expectation of $\bar{x}^*$ also equals $\sum_{i=1} x_i$, i.e., 1SMM yields an unbiased estimator for the sum of private inputs. We present the detailed privacy and utility analysis for 1SMM and dSMM later in Section 3.4.

3.3 Skellam Noise Preserves Privacy

Before we analyze the privacy guarantee of SMM, we first show that its building block, i.e., a single symmetric Skellam noise, preserves privacy, formalized as follows.

Theorem 3 (Rényi divergence of Skellam distributions). *For any integer $s \in \mathbb{Z}$ satisfying $|s| \leq \Delta_\infty$, any $\alpha > 1$, and any Δ_∞ satisfying $\alpha < 2\lambda/\Delta_\infty + 1$, we have*

$$D_\alpha(s + Sk(\lambda, \lambda) \,\|\, Sk(\lambda, \lambda)) \leq \frac{1.09\alpha + 0.91}{2} \cdot \frac{s^2}{2\lambda}. \tag{1}$$

We have the following multi-dimensional extension.

Theorem 4 (Rényi divergence of multi-dimensional Skellam distributions). *Let $Sk^d(\lambda, \lambda)$ denote a d-dimensional variate, where each dimension is independently sampled from $Sk(\lambda, \lambda)$. Then, for any integer-valued vector $s \in \mathbb{Z}^d$ satisfying $\|s\|_2^2 \leq c$ and $\|s\|_\infty \leq \Delta_\infty$, any $\alpha > 1$, and any Δ_∞ satisfying $\alpha < 2\lambda/\Delta_\infty + 1$, we have*

$$D_\alpha(s + Sk^d(\lambda, \lambda) \,\|\, Sk^d(\lambda, \lambda)) \leq \frac{1.09\alpha + 0.91}{2} \cdot \frac{c}{2\lambda}. \tag{2}$$

The proof of the above theorem can be found in Appendix C.1 of the technical report version [6]. Next, we highlight the contributions of our theoretical results. First, according to Theorem 3, the privacy guarantee provided by a symmetric Skellam noise of variance 2λ is comparable (i.e., within a constant factor) with that of adding continuous Gaussian noise of the same variance, which is $\frac{\alpha \cdot s^2}{2 \cdot 2\lambda}$ [36]. Further, as Eq. (1) only involves the quadratic term, the one-dimensional privacy analysis can be easily extended to the multi-dimensional setting by replacing the quadratic term with the squared $\mathcal{L}_2$ norm, as in Theorem 4.

Meanwhile, since additive Skellam noise preserves RDP, by Lemmata 1 and 2, it allows the tight privacy accounting of Skellam noises in applications involving composition and subsampling (e.g., FL), which is elaborated further in Section 4. Note that although our analysis restricts the value of Δ_∞ to $\Delta_\infty < 2\lambda/(\alpha - 1)$, this constraint only affects the utility of the Skellam noise, not its privacy guarantees. This is because the constraint can be easily enforced by standard $\mathcal{L}_\infty$ clipping. In addition, in the federated learning setting, λ is usually much larger than the optimal α (order of RDP) due to the fact that a large number of participants contribute to the overall DP noise, and the optimal α is often relatively small (e.g. less than 10 in our experiments). Hence, the above constraint leads to a sufficiently large range for $\mathcal{L}_\infty$ clipping without causing much utility degradation.

A notable difference between our theoretical result presented in Theorem 4 and the analysis of Skellam noise in [3] is that our result is "cleaner" in the sense that Eq. (2) only involves the $\mathcal{L}_2$ norm (similar to the case of continuous Gaussian noise [36]), whereas the analysis in [3] also involves the $\mathcal{L}_1$ norm of vector s. In general, the presence of $\mathcal{L}_1$ sensitivity may lead to an excessive amount of noise for high dimensional data, as the $\mathcal{L}_1$ sensitivity can be $\sqrt{d}$ times larger than the $\mathcal{L}_2$ sensitivity, limiting the applicability of Skellam noises in such applications. Further, the clean bound without the $\mathcal{L}_1$ norm term may also significantly simplify the design of protocols and mechanisms built on top of additive Skellam noises, e.g., algorithms 1SMM and dSMM presented earlier. To avoid the $\mathcal{L}_1$ norm term, we do not use known properties of Rényi divergence, and instead attack the problem directly using basic mathematical tools, which is a novel proving technique of independent interest (presented in detail in Appendix C.1 in the technical report version [6]). In particular, this proving technique leads to long and heavy formulae at the beginning, and yet within a few steps, most terms are canceled out, resulting in a clean bound.

Finally, we mention that there exists an exact sampler for the Skellam distribution, described Appendix A of the technical report version [6]). As a result, adding Skellam noise strictly preserves differential privacy. On the contrary, we are not aware of an exact sampler for the continuous Gaussian distribution. Consequently, the random noise sampled with an inexact sampler only approximately follows the Gaussian distribution; strictly speaking, injecting such noise may violate differential privacy, which is yet another motivation for employing our proposed method that injects Skellam noise.

3.4 Theoretical Analysis of Skellam Mixture Mechanism

We present the theoretical analysis of the proposed Skellam mixture mechanism (SMM). The proofs are deferred to Appendix C in the technical report version [6]. In terms of privacy, we have the following theorem for Algorithm 1.

Theorem 5. *Suppose that each participant's data point x_i satisfies*

$$|x_i|^2 + (|x_i| - \lfloor |x_i| \rfloor) - (|x_i| - \lfloor |x_i| \rfloor)^2 \leq c$$

and $\lceil |x_i| \rceil \leq \Delta_\infty$. Then, whenever $\alpha > 1$ and Δ_∞ satisfies

$$\alpha < \frac{2n\lambda}{\Delta_\infty} + 1, \text{ and } (10.9\alpha^2 - 1.8\alpha - 9.1) < \frac{4n\lambda}{\Delta_\infty^2}, \tag{3}$$

Algorithm 1 with noise parameter λ satisfies (α, τ)-RDP with $\tau = \frac{1.2\alpha+1}{2} \cdot \frac{c}{2n\lambda}$.

We extend Theorem 5 to the multi-dimensional setting using Lemma 1.

Corollary 1. *Suppose that each participant's data point x_i is d-dimensional and satisfies*

$$\sum_{j=1}^{d}\left(|x_{i,j}|^2 + \left(|x_{i,j}| - \lfloor |x_{i,j}| \rfloor\right) - \left(|x_{i,j}| - \lfloor |x_{i,j}| \rfloor\right)^2\right) \leq c, \quad (4)$$

and $\|\lceil |x_i| \rceil\|_\infty \leq \Delta_\infty$. Then, whenever $\alpha > 1$ and Δ_∞ satisfies Eq. (3), Algorithm 2 with noise parameter λ satisfies (α, τ)-RDP with $\tau = \frac{1.2\alpha+1}{2} \cdot \frac{c}{2n\lambda}$.

In practice, the constraints in Eq. (4) and $\|\lceil |x_i| \rceil\|_\infty \leq \Delta_\infty$ can be enforced by clipping, as we explain in Section 4. The maximum value of the $\mathcal{L}_\infty$ clipping bound Δ_∞ is computed from Eq. (3). Next, we present the utility guarantee incurred by Algorithm 2, which follows from Corollary 1.

Corollary 2. *Suppose that each participant's data point x_i is d-dimensional and satisfies Eq. (4), $\|\lceil |x_i| \rceil\|_\infty \leq \Delta_\infty$, $\alpha > 1$, and Δ_∞ satisfies Eq. (3). Then, when satisfying (α, τ)-RDP, the error incurred by Algorithm 1 is*

$$Err_{\mathcal{M}} = \frac{1.2\alpha + 1}{2} \cdot \frac{dc}{\tau} + \sum_{i=1}^{n}\sum_{j=1}^{d}\left(|x_{i,j}| - \lfloor |x_{i,j}| \rfloor - \left(|x_{i,j}| - \lfloor |x_{i,j}| \rfloor\right)^2\right).$$

We briefly comment on Corollary 2. We define $p_{i,j} := |x_{i,j}| - \lfloor |x_{i,j}| \rfloor$, which is the probability of increasing the absolute value $|x_{i,j}|$ by 1 for the i-th participant. Then, the overall error incurred by dSMM is:

$$Err_{\mathcal{M}} = \frac{(1.2\alpha + 1) \cdot dc}{2\tau} + \sum_{i=1}^{n}\sum_{j=1}^{d}(p_{i,j} - p_{i,j}^2).$$

The first term of $Err_{\mathcal{M}}$ can be viewed as the error due to enforcing differential privacy. Note that the leading multiplier $(1.2\alpha + 1)/2$ is only slightly larger (i.e., by a constant factor) than of the approach injecting continuous Gaussian noise, which is $\alpha/2$. The second error term is the overall variance of all the Bernoulli trials performed on the participant side. This error term can be seen as the integer approximation error, which exists even without enforcing differential privacy.

4 FEDERATED LEARNING WITH SKELLAM MIXTURE MECHANISM

In this section, we apply our Skellam mixture mechanism (SMM) to enforce DP on federated learning with distributed SGD. We assume that the participants have access to a black-box secure aggregation protocol, following the convention in [4, 28]. The training process is outlined in Algorithm 3. In each iteration, the server releases the current model parameters to all participants (Line 2 in Algorithm 3). Then, a random subset of participants, whose identities are not known to the server, is selected (Line 3). Each participant in the selected subset then computes the gradients based on the current model weights and her own data (Line 5), and invoke Algorithm 4 for gradient perturbation (Line 6). After that, the secure aggregation protocol computes the sum of the perturbed gradients (Line 7) of the randomly selected participants. Finally, the server retrieves the perturbed gradient sum and updates the model (Lines 8 and 9). We

Algorithm 3: Federated learning with Skellam mixture mechanism

Input: Private dataset of training records $X = (x_1, \ldots, x_n)$; initial model parameters θ; secure aggregation protocol $\mathcal{A}$.
Parameters: Sampling parameter q; number of iterations T; noise parameter λ; scale parameter γ; clipping thresholds c and Δ_∞; modulus $m \in \mathbb{N}$.

1 **for** $h \in 1 \ldots T$ **do**
2 The server shares the current model parameters θ to all participants.
3 $B \xleftarrow{u.a.r} \{1, 2, \ldots, n\}$. // sample a subset of participants uniformly at random from all participants using Poisson sampling with rate q
4 **for** $i \in B$ **do**
5 $g_i \leftarrow \nabla_\theta(r_i)$. // gradient computation
6 $z_i \leftarrow$ Algorithm 4(g_i). // SMM on the participant side
7 $\bar{z} \leftarrow \mathcal{A}(\{z_i\}_{i \in B})$. // secure aggregation
8 $\bar{g}^* \leftarrow$ Algorithm 6$(\bar{z})$. // gradient sum retrieval by the server
9 $\theta \leftarrow Update(\theta, \bar{g}^*)$. // model update based on the approximate gradient sum

Output: θ model parameters learnt on X.

Algorithm 4: participant procedure for perturbing gradients

Input: Private gradient $g_i \in \mathbb{R}^d$
Parameters: Noise parameter λ; scale parameter γ; clipping thresholds c and Δ_∞; modulus $m \in \mathbb{N}$.
Public randomness: Uniformly random sign vector $\xi \in \{-1, +1\}^d$.

1 $g_i \leftarrow H_d D_\xi g_i$. // random rotation, where $H \in \{-1/\sqrt{d}, +1/\sqrt{d}\}^{d \times d}$ is a Walsh-Hadamard matrix satisfying $H^T H = I$ and $D_\xi \in \{-1, 0, +1\}^{d \times d}$ is a diagonal matrix with ξ on the diagonal
2 $g_i \leftarrow \gamma \cdot g_i$. // scaling
3 $g_i \leftarrow clip(g_i)$. // clip g_i as in Algorithm 5
4 **for** $j \in 1 \ldots d$ **do**
5 $p_{i,j} = g_{i,j} - \lfloor g_{i,j} \rfloor$.
6 Sample $y_{i,j}$ from a Bernoulli trial with success probability $p_{i,k}$.
7 **if** $y_{i,j} = 0$ **then**
8 $g^*_{i,j} \leftarrow \lfloor g_{i,j} \rfloor + Sk(\lambda, \lambda)$.
9 **else**
10 $g^*_{i,j} \leftarrow \lfloor g_{i,j} \rfloor + 1 + Sk(\lambda, \lambda)$.
11 $z_{i,j} \leftarrow g^*_{i,j} \mod m$.

Output: $z_i \in \mathbb{Z}_m^d$ for the secure aggregation protocol.

omit additional details on the updating process (*e.g.*, learning rate schedule, weight decay) as they do not affect the general framework or the privacy guarantees. After repeating the above process for T iterations, the training terminates, and the server obtains the final model weights θ.

In what follows, we explain the participant procedure for perturbing gradients (Algorithm 4) and the server procedure for reconstructing the perturbed gradient sum (Algorithm 6). Each participant i first randomly rotates the private vector using a Walsh-Hadamard matrix [26] and a public random sign vector ξ shared

Algorithm 5: participant procedure for clipping gradients

Input: Private gradient $g_i \in \mathbb{R}^d$

Parameters: Clipping thresholds c and Δ_∞.

1 $v_i \leftarrow \mathbf{0}$. // initialize the helper vector for clipping.

2 **for** $j \in 1..d$ **do**

3 $\quad v_{i,j} = \frac{g_{i,j}}{|g_{i,j}|} \cdot \left(|g_{i,j}|^2 + |g_{i,j}| - \lfloor |g_{i,j}| \rfloor + (|g_{i,j}| - \lfloor |g_{i,j}| \rfloor)^2\right)$. // map g_i to v_i.

4 $v_i \leftarrow \min(1, \frac{c}{\|v_i\|_1}) \cdot v_i$. // $\mathcal{L}_1$ clip and re-scale

5 **for** $j \in 1 \ldots d$ **do**

6 $\quad g'_{i,j} = \lfloor \sqrt{|v_{i,k}|} \rfloor$. // compute the integer part

7 $\quad p'_{i,j} = \frac{y}{2g'_{i,j}+1}$ // compute the fraction part

8 $\quad g_{i,j} \leftarrow \frac{v_{i,j}}{|v_{i,j}|} \cdot (g'_{i,j} + p'_{i,j})$. // compose two parts

9 **for** $j \in 1 \ldots d$ **do**

10 $\quad g_{i,j} \leftarrow \frac{g_{i,j}}{|g_{i,j}|} \cdot \min(\Delta_\infty, |g_{i,j}|)$. // $\mathcal{L}_\infty$ clip.

Output: g_i the clipped gradient.

Algorithm 6: Server procedure of estimating gradient sum

Input: Private vector $\bar{z} = (\sum_{i \in B} z_i \mod m) \in \mathbb{Z}_m^d$ via secure aggreagtion

Parameters: Noise parameter λ; scale parameter γ; clipping thresholds c and Δ_∞; modulus $m \in \mathbb{N}$.

Public randomness: Uniformly random sign vector $\xi \in \{-1, +1\}^d$.

1 Map $\bar{z} \in \mathbb{Z}_m^d$ to $\bar{z}' \in [-m/2, m/2]^d \cap \mathbb{Z}^d$.

2 $\bar{g}^* \leftarrow \frac{1}{\gamma} \cdot D_\xi H_d^T \bar{z}'$.

Output: $\bar{g}^*$ the estimated gradient sum.

among all participants and the server (Line 1 in Algorithm 4), which is also used in previous solutions [3, 4, 28]. Each dimension of the resulting gradient follows a Sub-Gaussian distribution with variance $O(\|g_i\|_2^2/d)$, where g_i is the participant's private gradient value. Specifically, each dimension is concentrated around 0 when d is large, e.g., tens of thousand for neural networks. Essentially, this operation flattens the gradient and limits the probability of overflowing when computing the sum of gradients. We refer the reader to [4, 28] for detailed discussions.

After that, the participant scales the rotated vector and clips the scaled vector (Lines 2 and 3). We will explain the clipping procedure shortly (outlined in Algorithm 5). Then, for each k-th coordinate in the clipped vector, the participant samples one bit from the Bernoulli distribution of success probability $g_{i,k} - \lfloor g_{i,k} \rfloor$, where $g_{i,k}$ is the k-th element of the rotated vector g_i (Lines 5 and 6 in Algorithm 4). If the Bernoulli trial fails, the participant samples a noise following the Skellam distribution $Sk(\lambda, \lambda)$ and shift the outcome to $\lfloor g_{i,k} \rfloor$ (Lines 7 and 8); otherwise, the participant shifts the same outcome to $\lceil g_{i,k} \rceil$ (Lines 9 and 10). Finally, the participant applies element-wise modulo operation on the noisy vector (Line 11). Essentially, this step restricts the output from the participant to $\mathbb{Z}_m^d$, and enforces a $\log_2 m$-bit communication constraint per dimension, both of which are required by the secure aggregation protocol.

Next, the participants collectively compute the sum of their output noisy vectors through a secure aggregation protocol, and reveal the sum $(\sum_{i=1}^n z_i \mod m)$ to the server. As we have mentioned, parameter m can be seen as the per dimension communication for secure aggregation protocol. Although a larger m helps preserve information on the noisy gradients, such an m increases the communication cost, slowing down the aggregation process (especially with a communication-intensive secure aggregation protocol) as well as the model training overall. The problem is exacerbated when the participant is a mobile device with metered Internet connection. Hence, in practice it is often preferable to set a relatively small m, e.g., 2^8 in our experiments, which is equivalent to a communication constraint of one-byte per dimension.

After obtaining the sum $(\sum_{i=1}^n z_i \mod m)$, the server first unwraps the modulo operation (Line 1 in Algorithm 6). In particular, values in $\{m/2, m/2+1, \ldots, m-1\}$ are mapped back to $\{-m/2, -m/2+1, \ldots, -1\}$, respectively; and values in $\{0, 1, \ldots, m/2-1\}$ remain unchanged. This is because in Line 11 in Algorithm 4, values in $\{-m/2, -m/2+1, \ldots, -1\}$ are mapped to $\{m/2, m/2+1, \ldots, m-1\}$, respectively; and values in $\{0, 1, \ldots, m/2-1\}$ are mapped to themselves. We refer the reader to [28] for a detailed discussion on this issue. Then, the server reverses the rotation and scaling performed on the participant side (Line 2 in Algorithm 6), obtaining an unbiased estimate for the gradient sum.

Next, we explain the clipping procedure outlined in Algorithm 5. Recall that clipping is a standard step introduced in DPSGD [2] to bound the sensitivity of deep learning with DP. The clipping procedure in this work is slightly different from that in DPSGD, which clips the $\mathcal{L}_2$ norm of the gradient. The difference is due to the different privacy guarantee of SMM (see Theorem 5 and Corollary 1). Recall that the privacy guarantee of d-dimensional SMM relies on the following property of the input data g_i:

$$\lceil |g_i| \rceil \le \Delta_\infty, \text{ and}$$

$$\sum_{j=1}^{d} \left(|g_{i,j}|^2 + \left(|g_{i,j}| - \lfloor |g_{i,j}| \rfloor\right) - \left(|g_{i,j}| - \lfloor |g_{i,j}| \rfloor\right)^2 \right) \le c.$$

Accordingly, this requires a different clipping procedure. The first property is easy to enforce. For example, for $\Delta_\infty = 1$ and $x_i = -1.9$, we simply increase the x_i to -1. The second property is more complicated to enforce, as we explained next. For each participant, we first construct a helper vector v_i. In particular, each dimension of v_i is computed as follows:

$$v_{i,j} = \frac{g_{i,j}}{|g_{i,j}|} \cdot \left(|g_{i,j}|^2 + |g_{i,j}| - \lfloor |g_{i,j}| \rfloor + (|g_{i,j}| - \lfloor |g_{i,j}| \rfloor)^2 \right),$$

for $j = 1 \ldots, d$ (Line 3 in Algorithm 5). For completeness, we define $\frac{0}{0} = 1$. Next, we clip vector v_i based on its $\mathcal{L}_1$ norm in the standard way (Line 4 in Algorithm 5). Finally, we re-map the clipped vector to its original form (Lines 5 to 8 in Algorithm 5) and clip each dimension of the vector by Δ_∞ (Line 10 in Algorithm 5).

4.1 Privacy Analysis

In this section, we analyze the privacy guarantee of Algorithm 3. Observe that each iteration of Algorithm 3 can be seen as running the Skellam mixture mechanism on a random subset of gradients. This is because none of the model sharing (Line 2 in Algorithm 3), gradient sum reconstruction (Line 8 in Algorithm 3), or model updating (Line 9 in Algorithm 3) procedures incurs any additional privacy loss, as the updated model can be reconstructed by the

constructed perturbed gradient sum, which, in turn, can be computed from the perturbed gradient sum released from the secure aggregation protocol. In addition, since the identities of the random subset of participants are not known to the server, the privacy guarantee benefits from amplification by subsampling. (We refer the reader to [28] for a detailed discussion on this issue.) Hence, the privacy guarantee of Algorithm 3 follows by applying the composition (Lemma 1) and the amplification (Lemma 2) results to the privacy analysis of SMM (Corollary 1). A formal statement of the privacy guarantees of Algorithm 3 is as follows.

Theorem 6 (Privacy guarantee of Algorithm 3). *For sampling parameter q; sampled subset B; number of iterations T; noise parameter λ; and clipping thresholds c and Δ_∞, for any $\alpha > 1$ and Δ_∞ satisfies*

$$\alpha < \frac{2|B|\lambda}{\Delta_\infty} + 1, \text{ and } (10.9\alpha^2 - 1.8\alpha - 9.1) < \frac{4|B|\lambda}{\Delta_\infty^2}, \tag{5}$$

Algorithm 3 satisfies (α, τ)-RDP with

$$\tau = T \cdot \frac{1}{\alpha - 1} \cdot \log\left((1-q)^{\alpha-1}(\alpha q - q - 1) + \sum_{l=2}^{\alpha} \binom{\alpha}{l} (1-q)^{\alpha-l} q^l e^{(l-1)\tau(l)}\right),$$

where $\tau(l)$ is defined as $\tau(l) := \frac{1.2l+1}{2} \cdot \frac{c}{2|B|\lambda}$, for $l = 2, \ldots, \alpha$.

5 RELATED WORK

As mentioned in Section 1, existing work on federated learning with differential privacy has mostly considered the non-MPC settings where real-value noise can be used. To our knowledge, there are only four prior studies [3, 4, 28, 30] on using integer noise to achieve DP in federated learning. In what follows, we revisit the solutions in [3, 4, 28, 30], and compare them with our SMM.

cpSGD [4]. cpSGD lets each participant inject binomial noise (*i.e.*, the sum of multiple binary values drawn from independent Bernoulli trials) to her discretized gradients to satisfy DP. Similar to Gaussian noise in the continuous domain, binomial noise can also be easily aggregated, as the sum of multiple i.i.d. binomial values also follows a binomial distribution. This property simplifies the privacy reasoning for cpSGD, as it allows us to focus on the aggregated binomial noise in the sum of all participants' gradients, without the need to analyze each participant's binomial noise separately. However, the privacy analysis of cpSGD in [4] is based on (ϵ, δ)-DP instead of RDP, which leads to relatively loose privacy bounds for federated learning, since it is difficult to derive the exact (ϵ, δ)-DP guarantee of an iterative algorithm with subsampling (e.g., SGD).

Another limitation of cpSGD is that it assumes the input to to be integer-valued. For any non-integer input x, the method requires a *stochastic rounding* [4] of x, which often leads to a considerable increase in sensitivity. For example, if $x = \{0.01, 0.01, \ldots, 0.01\} \in \mathbb{R}^d$, then each dimension of x is rounded 1 with 0.01 probability, and to 0 with 0.99 probability. This approach ensures that each rounded value's expectation equals the original value, but the rounded values could be significantly larger than the original ones. In particular, in the worst case when each dimension of x is rounded to 1, the $\mathcal{L}_2$ norm of x is increased from $0.01 \cdot \sqrt{d}$ to $\sqrt{d}$ after the rounding. In other words, even if each participant's gradient vector has an $\mathcal{L}_2$ norm at most $0.01 \cdot \sqrt{d}$, the sum of all participant's rounded gradients could have an $\mathcal{L}_2$ sensitivity of $\sqrt{d}$. This significantly increases the amount of noise required by cpSGD to achieve differential privacy, resulting in an unfavorable trade-off between privacy and utility.

Distributed Discrete Gaussian (DDG) mechanism [28]. To mitigate the limitations of cpSGD, Kairouz *et al.* [28] propose DDG, a method that utilizes *discrete Gaussian distributions* [11] instead of binomial distributions for noise generation. In particular, a discrete Gaussian distribution has a similar PDF to a continuous Gaussian distribution, but is defined over the integer domain. The main advantage of using discrete Gaussian noise is that it can achieve RDP, which makes it much easier to derive a tight privacy bound of DDG for iterative algorithms with subsampling.

Similar to cpSGD, DDG also assumes that the inputs are integer-valued, and, hence, requires stochastic rounding of non-integers. To alleviate the sensitivity increase incurred by rounding, DDG applies a *conditional rounding* approach as follows. First, given an input $x \in \mathbb{R}^d$ with bounded $\mathcal{L}_2$ norm Δ_2 (otherwise DDG clips the input) and the scale parameter γ, DDG scales the input x and obtains γx. After that, DDG performs a stochastic rounding on γx. If the rounded version of the scaled input has an $\mathcal{L}_2$ norm larger than

$$\sqrt{\gamma^2\Delta_2^2 + d/4 + \sqrt{2\log(1/\beta)}(\gamma\Delta_2 + \sqrt{d}/2)}, \tag{6}$$

for some fixed β (explained soon), then DDG discards it and regenerates another stochastically rounded version. The procedure is repeated until the above requirement is met. The hyperparameter β ranging from 0 to 1 controls the trade-off between bias and sensitivity increase in the conditional rounding process. To see this, note that the expectation of the rounded value is generally not equal to the original value (since rounded values failing the above condition in Eq. (6) are rejected), which adversely affects the accuracy of the output of DDG. A smaller β leads to a lower bias but higher sensitivity increase, which, in turn, leads to a higher amount of noise needed to satisfy DP, and vice versa. This conditional rounding approach ensures that the rounding operation does not incur a significant increase of the $\mathcal{L}_2$ sensitivity, but the increase is still $O(\sqrt{d})$. In addition, the conditional rounding operation introduces a hyperparameter β, which is difficult to tune under DP. The authors of [28] recommend fixing β to $e^{-0.5}$, which is done in our experiments.

Skellam Mechanism [3]. In Ref. [3], Agarwal *et al.* propose to sample noise from a Skellam distribution instead of a discrete Gaussian distribution. Since the sum of independent Skellam noises still follows Skellam distribution (see Section 2.1), the privacy reasoning of distributed Skellam noise is straightforward, unlike DDG. In particular, the paper shows that adding Skellam-distributed noise to integers also achieves RDP. However, for non-integer inputs, the Skellam mechanism in [3] still requires the conditional rounding approach introduced in [28]. Consequently, its accuracy also suffers from the sensitivity increase, as well as the bias introduced by conditional rounding.

Comparisons with SMM. Compared to the aforementioned methods, one major advantage of SMM is that it does not rely on an additional stochastic rounding [4] or conditional rounding [3, 28] step to handle non-integer inputs. Instead, SMM directly takes any $x \in \mathbb{R}^d$ as input, and outputs a noisy version x^* of x whose expectation equals x, without incurring a significant increase in sensitivity. Accordingly, SMM injects a smaller amount of noise while achieving the same level of privacy as its competitors. As a consequence, SMM is able to obtain much more accurate results than cpSGD [4], DDG [28], and the Skellam mechanism [3], especially in settings where communication is constrained to low bitwidths. In particular, in such situations, the quantization granularity is set to a coarse level (i.e., a small scale parameter γ) to avoid overflow. Such a coarse quantization granularity leads to a relatively large sensitivity increase compared to the quantized gradients. For this reason, the perturbation noise in cpSGD, DDG, and Skellam due to rounding is rather high in such low-bitwidth settings, resulting in much lower model utility than SMM. We validate this claim with experiments in the next section.

In addition, compared with DPSGD [2], SMM involves only one additional hyper parameter: the scale parameter γ, which controls the trade-off between communication cost and utility. Note that this trade-off does not exist in DPSGD as it is a solution for the centralized setting. Once γ is determined, we can compute the clipping threshold c for SMM as $c = \gamma^2 \cdot \Delta_2^2$ for some constant Δ_2, which corresponds to setting the $\mathcal{L}_2$ clipping norm to Δ_2 in DPSGD [2]. In addition, the $\mathcal{L}_\infty$ clipping bound Δ_∞ for SMM is computed from Eq. (5). In contrast, both DDG [28] and the Skellam mechanism [3] include an additional hyperparameter β. In these algorithms, parameter β controls the trade-off between bias and sensitivity in their conditional rounding process, as mentioned earlier. A poor choice of β may adversely impact the performance of these algorithms; meanwhile, hyperparameter tuning is rather challenging under the differential privacy requirement.

We also note that our theoretical analysis of SMM is substantially different from that in [3], due to the inherent differences between the Skellam mixture distribution used in SMM and the Skellam distribution used in [3]. In addition, even for the special case of integer inputs, the privacy guarantee of SMM (see Theorems 3 and 4) differs from that of the Skellam mechanism in [3], because we use different proof techniques from those in [3]. While the techniques used in [3] is also non-trivial, our result for integer inputs is cleaner, and is of independent interest, as we have mentioned in Section 3.3.

6 EXPERIMENTS

We evaluate the performance of SMM on the distributed sum estimation problem and two basic machine learning tasks. For simplicity, all experiments are done using the approximate samplers for Discrete Gaussian and Skellam from the TensorFlow libraries, which are based on floating point approximations. Compared with exact samplers, approximate samplers are faster. We include a detailed discussion on this issue in Appendix A in the technical report version [6].

6.1 Distributed Sum Estimation

As a simple application, we first evaluate the performance of solution SMM on the distributed sum estimation problem described in Section 3.1, given a private d-dimensional input dataset. Following the experiment setting in [28], we generate a synthetic dataset containing $n = 100$ data points uniformly sampled from a d-dimensional $\mathcal{L}_2$ sphere. We set the dimension to $d = 65536$, and the radius to $r = 1$ (namely, the $\mathcal{L}_2$ sensitivity of input is 1). The participants release their noisy sum under distributed DP. We report the mean squared error (mse) over all dimensions. Our evaluation uses the (ϵ, δ)-DP (Definition 2) definition instead of RDP (Definition 3, since (ϵ, δ)-DP is a classic definition of differential privacy, and a competitor cpSGD supports the former but not the latter. We fix δ to 10^{-5}, and vary the privacy parameter ϵ from $\{1, 2, 3, 4, 5\}$. For DDG, Skellam, and SMM, we first compute the privacy guarantee using RDP, and then convert the guarantee to (ϵ, δ)-DP using Lemma 3 (the optimal RDP order is chosen from integers from 2 to 100).

For DDG, Skellam, cpSGD, and SMM, we vary the communication bitwidth per dimension from $\{10, 12, 14, 16, 18\}$. Correspondingly, m varies from $\{2^{10}, 2^{12}, 2^{14}, 2^{16}, 2^{18}\}$ (see line 11 in Algorithm 4). For m equals to $2^{10}, 2^{12}, 2^{14}, 2^{16}$, and 2^{18}, we vary the scale parameter γ in $\{4, 8\}$, $\{16, 32\}$, $\{64, 128\}$, $\{256, 512\}$, and $\{1024, 2048\}$, respectively (see line 2 in Algorithm 4). For SMM, the clipping threshold c is set to $\gamma^2 r^2$ with $r = 1$. Additionally, we compute the $\mathcal{L}_\infty$ clipping bound for SMM using Eq. (3), based on the optimal RDP order. For Skellam and DDG, the $\mathcal{L}_2$ clipping bound is set to $\Delta_2 = \sqrt{\gamma^2 r^2 + d/4 + \sqrt{2\log(1/\beta)}(\gamma r + \sqrt{d}/2)}$, with $r = 1$, $d = 65536$, and $\beta = \exp(-0.5)$, as suggested in [28]. In terms of $\mathcal{L}_1$ clipping bound for Skellam and DDG, we have that $\Delta_1 \leq \min\left(\sqrt{d} \cdot \Delta_2, \Delta_2^2\right)$, following [28]. Note that we do not perform an actual $\mathcal{L}_1$ clipping step for the rounded gradients, as the above relationship between $\mathcal{L}_2$ and $\mathcal{L}_1$ norms automatically holds for all integer-valued vectors. Similarly, for cpSGD, the $\mathcal{L}_1$ norm is bounded by $\sqrt{d}$ times the $\mathcal{L}_2$ norm, following its original implementation. We also include continuous Gaussian, which is a solution for the centralized DP setting, as a strong baseline.

The results are shown in Figure 1. When the communication bitwidth is limited (i.e., when $m = 2^{10}, 2^{12}, 2^{14}$), SMM significantly outperforms all its competitors, as demonstrated in Figures 1 (a), (b), (c), (f), (g), and (h). When $m = 2^{16}$ and $\gamma = 256$, SMM achieves comparable performance as DDG and Skellam, as shown in Figure 1 (d). When both m and γ are large, SMM performs slightly worse than DDG and Skellam, which obtain almost the same accuracy as the strong baseline continuous Gaussian, as we see from Figures 1 (i), (e), and (j). Finally, Skellam and DDG has similar performance under all settings, and cpSGD incurs rather high error ($> 10^4$), and falls outside the error range shown in the figures. Below, we briefly explain the reasons for the above results.

As mentioned in Section 5, existing solutions for distributed DP incur high sensitivity overhead due to stochastic rounding (in cpSGD) or conditional rounding (in DDG and Skellam). To be more specific, the sensitivity overhead is roughly 1 per dimension, which is non-negligible compared to the scaled data, especially when the data dimension is large (e.g., $d = 65536$) and when the quantization

granularity is coarse (i.e., small γ) under small bitwidths (i.e., small m). This sensitivity increase leads to stronger perturbations for cpSGD, DDG, and Skellam, and explains why SMM performs the best in settings with small bitwidths. As the bitwidth increases with γ, the above-mentioned sensitivity overhead becomes negligible compared with the scaled data. As a result, Skellam and DDG yield almost the same accuracy as continuous Gaussian. In the meantime, SMM performs slightly worse than continuous Gaussian, DDG, and Skellam. This is because SMM always incurs a slightly larger error than continuous Gaussian, according to Corollary 2, where there is an extra factor of 1.2 leading the error term of SMM.

6.2 Federated Learning

Next, we evaluate the performance of the proposed solution SMM on FL with DP (Algorithm 3) on two classic benchmark datasets: MNIST [31] and Fashion MNIST [49], which contain grayscale images of handwritten digits and clothing, respectively. Both datasets represent 10-class classification tasks with $60,000$ training data records. We regard each data record in the training data as a participant. Our evaluation uses the (ϵ, δ)-DP, as we have explained earlier. We fix δ to 10^{-5}, and vary the privacy parameter ϵ from $\{1, 2, 3, 4, 5\}$. In particular, for cpSGD, we apply both linear composition and advanced composition [19] for privacy accounting and choose the stronger guarantee between them. We have also included the strong central-model DPSGD [2] as a baseline.

For both MNIST [31] and Fashion MNIST [49], we train a three-layer neural network with fully connected layers and ReLu activation, following previous work [4]. We set the number of neurons per layers to 80, resulting in a model with $d = 63,610$ weights. For DDG, Skellam, cpSGD, and SMM, we vary the communication constraint m from $\{2^6, 2^8, 2^{10}\}$, where $m = 2^8$ corresponds to one byte per parameter. For each m, we vary the scaling parameter γ in $\{m/32, m/16, m/8, m/4, m/2, m\}$ (see line 2 in Algorithm 4). For cpSGD, DDG, Skellam, and the centralized algorithm DPSGD, we use the same $\mathcal{L}_2$ clipping norm of 1 for the original real-valued gradients. For the scaled gradients in DDG and Skellam, we set $\mathcal{L}_2$ clipping bound to $\sqrt{\gamma^2\Delta_2^2 + d/4 + \sqrt{2\log(1/\beta)}(\gamma\Delta_2 + \sqrt{d}/2)}$, with $\Delta_2 = 1$, $d = 65536$, and $\beta = \exp(-0.5)$. For SMM, we set the clipping threshold c to $\gamma^2\Delta_2^2$, with $\Delta_2 = 1$, similar to its competitors. In terms of the $\mathcal{L}_\infty$ clipping bound for SMM, we compute Δ_∞ from Eq. (3) using the optimal order of α. We also vary batch size $|B|$ from $\{120, 240, 480, 960\}$. The model is trained for 4 epochs, i.e., when $|B|$ equals to $120, 240, 480$, and 960, we train the model for $2000, 1000, 500$, and 250 rounds, respectively. For all experiments, we use the Adam optimizer [29] with learning rate $\eta = 0.005$. We do not tune the hyper-parameters in favor of any particular solution and omit additional experiments on hyper parameter tuning, e.g., model structure, learning rate, clipping norm, optimizer, training epochs, etc. We remark that our approach is compatible with existing differentially private parameter tuning techniques [25, 32, 40], which is an orthogonal topic to this paper. We report the average test accuracy over 5 runs. The results are shown in Figures 2 and 3.

Overall, the results are consistent with the those for distributed sum estimation, and lead to similar conclusions as before, i.e., SMM has a clear performance advantage over its competitors with small bitwidths, and the performance gap gradually closes as the bitwidth increases.

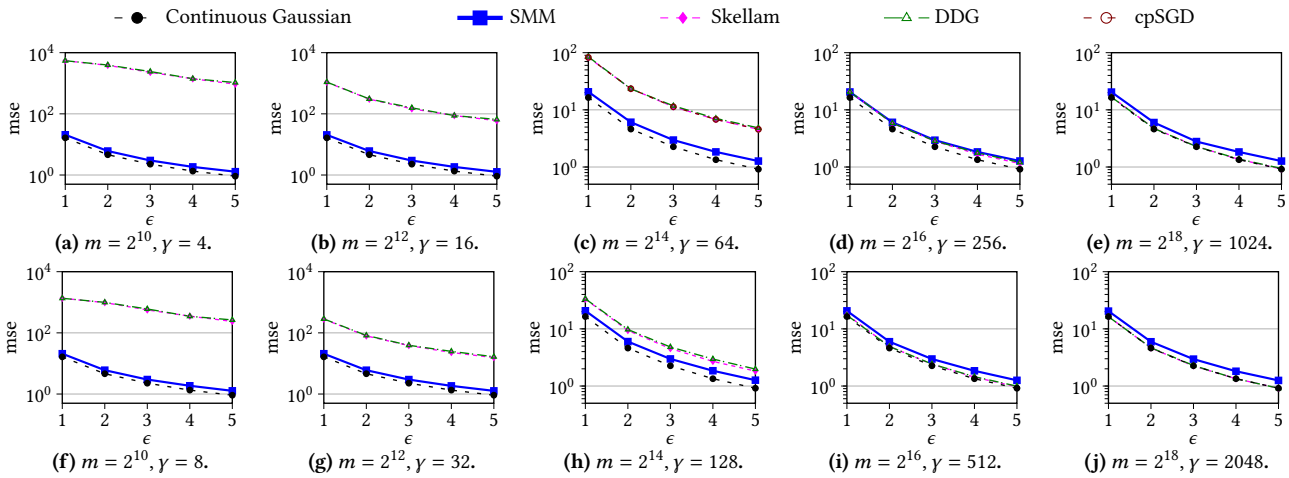
Specifically, when $m = 2^6$, SMM is the only method that achieves meaningful accuracy under all settings of privacy parameter ϵ, batch size $|B|$, and scale ratio γ (see Figures 2(a), (b), and (c), and Figures 3(a), (b), and (c)). This is because the scale of the noise injected in DDG, Skellam, and cpSGD is so large that it causes floating point number overflow, destroying the utility of the resulting gradient sum.

When $m = 2^8$ (i.e., one byte per parameter), SMM also achieves significantly higher accuracy compared to its competitors. In particular, in Figures 2(d) and 3(d), we fix the scale parameter to $\gamma = 64$ and the batch size to $|B| = 240$. When $\epsilon = 1$, DDG and Skellam yield very low utility due to floating point number overflows, while SMM achieves much higher utility that is close to that of DPSGD (i.e., the gap is less than 10%). As ϵ increases (indicating weaker privacy protection), the performance gap between SMM and its competitors becomes less dramatic. This is because with a higher ϵ, the required noise scale for the competitors becomes smaller, to the point that it no longer causes floating point number overflows. Nevertheless, there remains a noticable performance gap, since the noise scale of SMM is still significantly lower than that of its competitors. In particular, when $\epsilon = 3$, the accuracy improvement of SMM over DDG and Skellam is around 6% and 10% for MNIST and Fashion MNIST, respectively, while the accuracy gap between SMM and the centralized baseline DPSGD is only around 3%.

The performance gap between SMM and the centralized DPSGD algorithm exists, even when ϵ reaches as high as 5. Not that at this point, the noise required to satisfy DP no longer dominates the total amount of perturbations; instead, the relatively coarse quantization granularity (i.e., caused by a small γ) becomes a significant factor. As we demonstrate shortly, this accuracy gap gradually closes with a larger bitwidth and/or a large scale ratio γ.

In Figures 2(e) and 3(e), we fix the privacy parameter $\epsilon = 3$ and the scale parameter $\gamma = 64$, and vary the batch size $|B|$ from $\{120, 240, 480, 960\}$. SMM is the only algorithm that consistently achieves comparable accuracy with DPSGD under all settings of $|B|$. In particular, when $|B| = 960$, the accuracy improvement of SMM over DDG and Skellam is around 30% and 20% for MNIST and Fashion MNIST, respectively. Finally, we fix the privacy parameter and the batch size, and vary the scale parameter γ in Figures 2(f) and 3(f). The results show consistent accuracy improvement with varying γ. We also note that as γ increases from 8 to 256, the accuracy of SMM first increases then decreases. On the one hand, as γ increases, the gradient weights becomes more fine-grained and contains more information, leading to higher accuracy; on the other hand, as γ increases, a larger amount of noise is required to satisfy DP. When $\gamma = 256$, the noisy gradient weights exceed the one-byte communication constraint, causing utility degradation. The same performance pattern can be observed for Skellam and DDG.

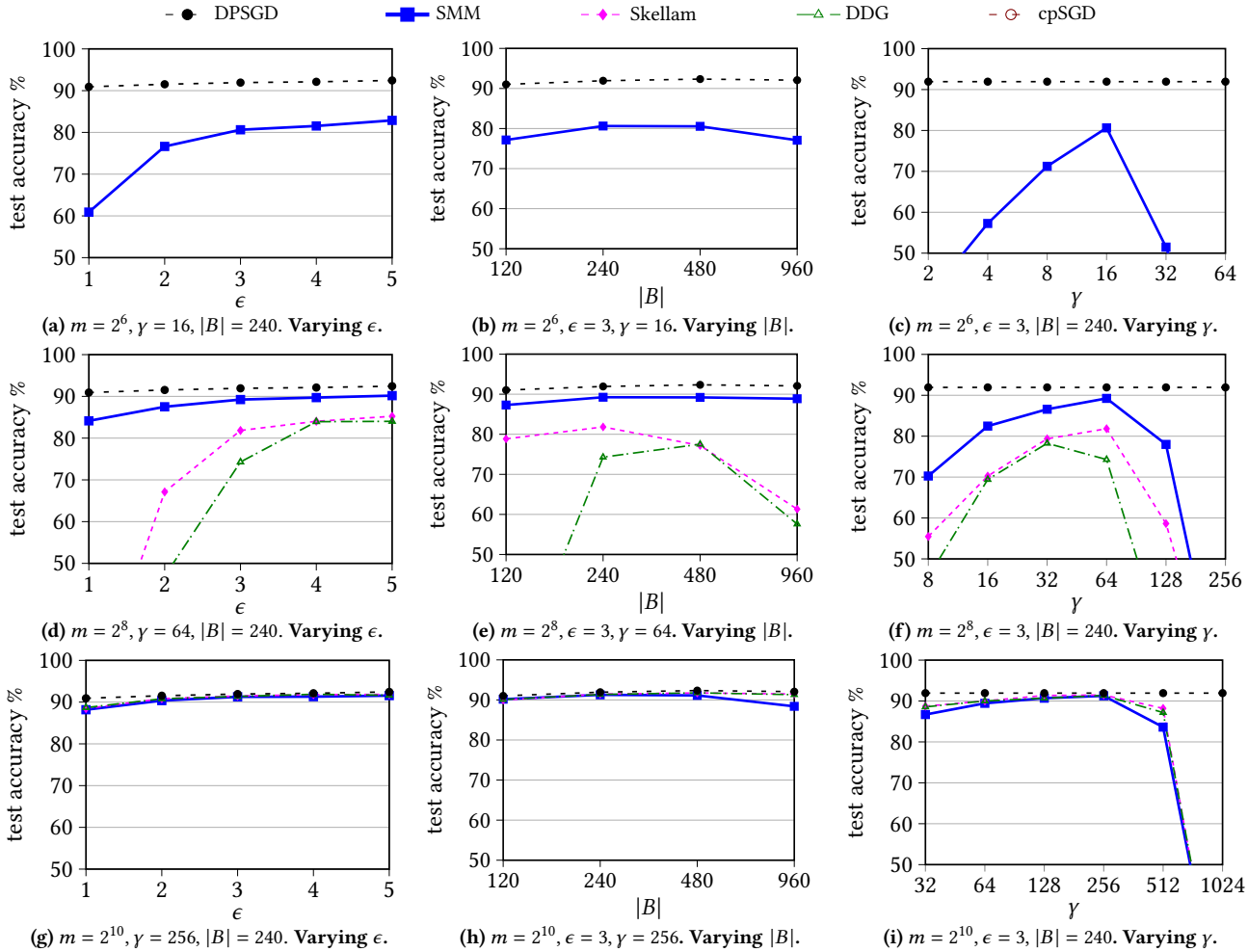
When the communication bitwidth is sufficiently large (e.g., when $m = 2^{10}$), we observe that while DDG and Skellam achieve almost the same accuracy as DPSGD, there is a small accuracy gap between SMM and DPSGD. For example, as we see in Figures 2(g) and 3(g), there is a 0.5 and 1 percent accuracy gap for MNIST and Fashion MNIST between SMM and DPSGD for $\epsilon \geq 2$. In addition, there are also noticeable accuracy gaps when $|B| = 960$



(a) $m = 2^{10}, \gamma = 4$. (b) $m = 2^{12}, \gamma = 16$. (c) $m = 2^{14}, \gamma = 64$. (d) $m = 2^{16}, \gamma = 256$. (e) $m = 2^{18}, \gamma = 1024$.

(f) $m = 2^{10}, \gamma = 8$. (g) $m = 2^{12}, \gamma = 32$. (h) $m = 2^{14}, \gamma = 128$. (i) $m = 2^{16}, \gamma = 512$. (j) $m = 2^{18}, \gamma = 2048$.

Figure 1: Evaluations on synthetic data with varying privacy parameter ϵ, scale parameter γ, and communication constraint m.

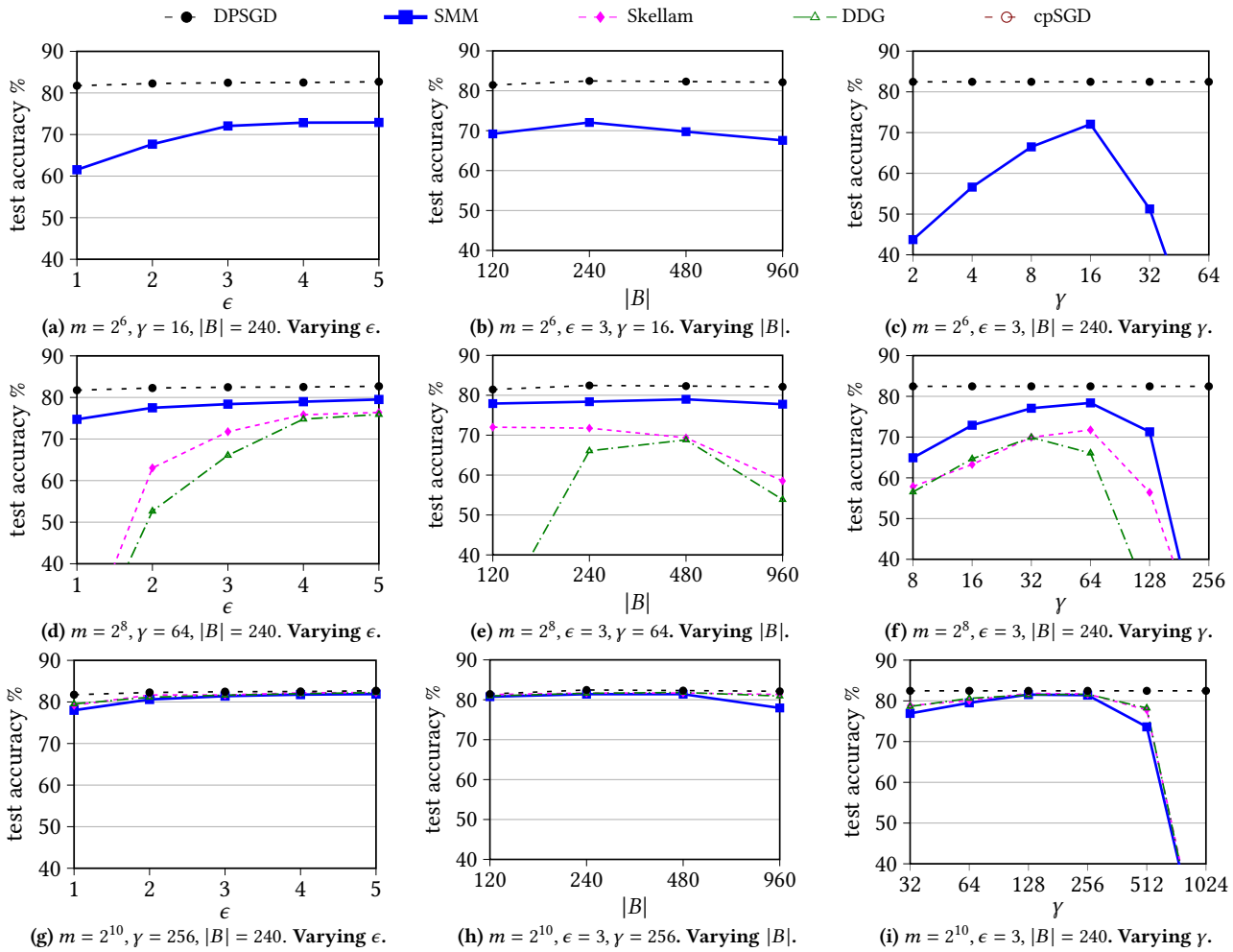


(a) $m = 2^6, \gamma = 16, |B| = 240$. **Varying** ϵ. (b) $m = 2^6, \epsilon = 3, \gamma = 16$. **Varying** $|B|$. (c) $m = 2^6, \epsilon = 3, |B| = 240$. **Varying** γ.

(d) $m = 2^8, \gamma = 64, |B| = 240$. **Varying** ϵ. (e) $m = 2^8, \epsilon = 3, \gamma = 64$. **Varying** $|B|$. (f) $m = 2^8, \epsilon = 3, |B| = 240$. **Varying** γ.

(g) $m = 2^{10}, \gamma = 256, |B| = 240$. **Varying** ϵ. (h) $m = 2^{10}, \epsilon = 3, \gamma = 256$. **Varying** $|B|$. (i) $m = 2^{10}, \epsilon = 3, |B| = 240$. **Varying** γ.

Figure 2: Evaluations on MNIST with varying communication constraint m, privacy parameter ϵ, scale parameter γ, and batch size $|B|$.



(a) $m = 2^6, \gamma = 16, |B| = 240$. **Varying** ϵ. **(b)** $m = 2^6, \epsilon = 3, \gamma = 16$. **Varying** $|B|$. **(c)** $m = 2^6, \epsilon = 3, |B| = 240$. **Varying** γ.

(d) $m = 2^8, \gamma = 64, |B| = 240$. **Varying** ϵ. **(e)** $m = 2^8, \epsilon = 3, \gamma = 64$. **Varying** $|B|$. **(f)** $m = 2^8, \epsilon = 3, |B| = 240$. **Varying** γ.

(g) $m = 2^{10}, \gamma = 256, |B| = 240$. **Varying** ϵ. **(h)** $m = 2^{10}, \epsilon = 3, \gamma = 256$. **Varying** $|B|$. **(i)** $m = 2^{10}, \epsilon = 3, |B| = 240$. **Varying** γ.

Figure 3: Evaluations on Fashion MNIST with varying communication constraint m, privacy parameter ϵ, scale parameter γ, and batch size $|B|$.

and when $\gamma = 512$ in Figures 2(h) and (i) and 3(h) and (i). Overall, as the bitwidth increases, the performance increase of DDG and Skellam is much more significant than SMM, whose performance is relatively stable with different bitwidths. Lastly, under all settings, the accuracy of cpSGD is rather low ($< 20\%$), and falls outside the accuracy range shown in the figures.

7 CONCLUSION

This paper presents the Skellam mixture mechanism (SMM), a novel solution for enforcing differential privacy on machine learning models built through an MPC-based federated learning process using distributed stochastic gradient descent. Compared to existing solutions, SMM achieves composable and scalable privacy guarantee without increasing the sensitivity of input. Extensive experiments, performed on both a synthetic dataset and two classic benchmark datasets, as well as various practical settings, demonstrate the consistent and significant accuracy gains SMM over existing solutions under restrictive communication constraints.

For future work, we plan to further reduce the constant factor in the privacy analysis for SMM to improve model utility under the same level of privacy protection. Another promising direction is to open up the black box of the MPC protocol and perform careful privacy analysis with considerations for the details of the MPC protocol, which might help lower the noise level further, leading to a more favorable privacy-utility trade-off for federated learning.

ACKNOWLEDGMENTS

This work was supported by the Ministry of Education, Singapore (Number MOE2018-T2-2-091), A*STAR, Singapore (Number A19E3b0099), and Qatar National Research Fund Qatar Foundation (Number NPRP11C-1229-170007). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not reflect the views of the funding agencies.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: A System for Large-Scale Machine Learning. In *OSDI*. 265–283.

[2] Martín Abadi, Andy Chu, Ian J. Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep Learning with Differential Privacy. In *CCS*. 308–318.

[3] Naman Agarwal, Peter Kairouz, and Ziyu Liu. 2021. The Skellam Mechanism for Differentially Private Federated Learning. In *NeurIPS*. 5052–5064.

[4] Naman Agarwal, Ananda Theertha Suresh, Felix Yu, Sanjiv Kumar, and H. Brendan McMahan. 2018. CpSGD: Communication-Efficient and Differentially-Private Distributed SGD. In *NeurIPS*. 7575–7586.

[5] Prabhanjan Ananth, Arka Rai Choudhuri, Aarushi Goel, and Abhishek Jain. 2018. Round-Optimal Secure Multiparty Computation with Honest Majority. In *CRYPTO*. 395–424.

[6] Ergute Bao, Yizheng Zhu, Xiaokui Xiao, Yin Yang, Beng Chin Ooi, Benjamin Hong Meng Tan, and Khin Mi Mi Aung. 2022. *Skellam Mixture Mechanism: a Novel Approach to Federated Learning with Differential Privacy (Technical report)*. Retrieved May 15, 2022 from https://drive.google.com/file/d/1k6HILAQC5_mwjFflQ-VJazBuRfrDurfU/view?usp=sharing

[7] Amos Beimel, Ariel Gabizon, Yuval Ishai, Eyal Kushilevitz, Sigurd Meldgaard, and Anat Paskin-Cherniavsky. 2014. Non-Interactive Secure Multiparty Computation. In *CRYPTO*. 387–404.

[8] James Henry Bell, Kallista A. Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. 2020. Secure Single-Server Aggregation with (Poly)Logarithmic Overhead. In *CCS*. 1253–1269.

[9] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *CCS*. 1175–1191.

[10] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *NeurIPS*. 1877–1901.

[11] Clément L. Canonne, Gautam Kamath, and Thomas Steinke. 2020. The Discrete Gaussian for Differential Privacy. In *NeurIPS*.

[12] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. 2019. The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks. In *SEC*. 267–284.

[13] David Chaum, Ivan Damgård, and Jeroen van de Graaf. 1987. Multiparty Computations Ensuring Privacy of Each Party's Input and Correctness of the Result. In *CRYPTO*. Springer, 87–119.

[14] Albert Cheu, Adam D. Smith, Jonathan R. Ullman, David Zeber, and Maxim Zhilyaev. 2019. Distributed Differential Privacy via Shuffling. In *EUROCRYPT*. 375–403.

[15] Adam Coates, Brody Huval, Tao Wang, David J. Wu, Bryan Catanzaro, and Andrew Y. Ng. 2013. Deep learning with COTS HPC systems. In *ICML*. 1337–1345.

[16] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. 2015. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press.

[17] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V. Le, Mark Z. Mao, Marc'Aurelio Ranzato, Andrew W. Senior, Paul A. Tucker, Ke Yang, and Andrew Y. Ng. 2012. Large Scale Distributed Deep Networks. In *NeurIPS*. 1232–1240.

[18] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *TCC*. 265–284.

[19] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Found. Trends Theor. Comput. Sci.* 9, 3-4 (2014), 211–407.

[20] Cynthia Dwork, Adam D. Smith, Thomas Steinke, Jonathan R. Ullman, and Salil P. Vadhan. 2015. Robust Traceability from Trace Amounts. In *FOCS*. 650–669.

[21] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. 2019. Amplification by Shuffling: From Local to Central Differential Privacy via Anonymity. In *SODA*. 2468–2479.

[22] Vitaly Feldman. 2020. Does Learning Require Memorization? A Short Tale about a Long Tail. In *STOC*. 954–959.

[23] Rosario Gennaro, Yuval Ishai, Eyal Kushilevitz, and Tal Rabin. 2002. On 2-Round Secure Multiparty Computation. In *CRYPTO*. 178–193.

[24] Slawomir Goryczka, Li Xiong, and Vaidy Sunderam. 2013. Secure Multiparty Aggregation with Differential Privacy: A Comparative Study. In *Joint EDBT/ICDT 2013 Workshops*. 155–163.

[25] Anupam Gupta, Katrina Ligett, Frank McSherry, Aaron Roth, and Kunal Talwar. 2010. Differentially Private Combinatorial Optimization. In *SODA*. 1106–1125.

[26] A. Hedayat and W. D. Wallis. 1978. Hadamard Matrices and Their Applications. *The Annals of Statistics* 6, 6 (1978), 1184 – 1238.

[27] Yuval Ishai, Eyal Kushilevitz, and Anat Paskin. 2010. Secure Multiparty Computation with Minimal Interaction. In *CRYPTO*. 577–594.

[28] Peter Kairouz, Ziyu Liu, and Thomas Steinke. 2021. The Distributed Discrete Gaussian Mechanism for Federated Learning with Secure Aggregation. In *ICML*. 5201–5212.

[29] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

[30] Antti Koskela, Joonas Jälkö, Lukas Prediger, and Antti Honkela. 2021. Tight Differential Privacy for Discrete-Valued Mechanisms and for the Subsampled Gaussian Mechanism Using FFT. In *AISTATS*. 3358–3366.

[31] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[32] Jingcheng Liu and Kunal Talwar. 2019. Private Selection from Private Candidates. In *STOC*. 298–309.

[33] Ryan T. McDonald, Keith B. Hall, and Gideon Mann. 2010. Distributed Training Strategies for the Structured Perceptron. In *HLT-NAACL*. 456–464.

[34] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *AISTATS*. 1273–1282.

[35] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. 2019. Exploiting Unintended Feature Leakage in Collaborative Learning. In *S&P*. 691–706.

[36] Ilya Mironov. 2017. Rényi Differential Privacy. In *CSF*. 263–275.

[37] Ilya Mironov, Kunal Talwar, and Li Zhang. 2019. Rényi Differential Privacy of the Sampled Gaussian Mechanism. *CoRR* abs/1908.10530 (2019).

[38] Milad Nasr, Reza Shokri, and Amir Houmansadr. 2019. Comprehensive Privacy Analysis of Deep Learning: Stand-alone and Federated Learning under Passive and Active White-box Inference Attacks. In *S&P*. 739–753.

[39] Pascal Paillier. 1999. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *EUROCRYPT*. 223–238.

[40] Nicolas Papernot and Thomas Steinke. 2021. Hyperparameter Tuning with Renyi Differential Privacy. *CoRR* abs/2110.03620 (2021).

[41] Apostolos Pyrgelis, Carmela Troncoso, and Emiliano De Cristofaro. 2018. Knock Knock, Who's There? Membership Inference on Aggregate Location Data. In *NDSS*.

[42] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *S&P*. 3–18.

[43] Congzheng Song, Thomas Ristenpart, and Vitaly Shmatikov. 2017. Machine Learning Models That Remember Too Much. In *CCS*. 587–601.

[44] Congzheng Song and Vitaly Shmatikov. 2019. Auditing Data Provenance in Text-Generation Models. In *KDD*. 196–206.

[45] Congzheng Song and Vitaly Shmatikov. 2020. Overlearning Reveals Sensitive Attributes. In *ICLR*.

[46] Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, Rui Zhang, and Yi Zhou. 2019. A Hybrid Approach to Privacy-Preserving Federated Learning - (Extended Abstract). *Inform. Spektrum* 42, 5 (2019), 356–357.

[47] Filipp Valovich and Francesco Aldà. 2017. Computational Differential Privacy from Lattice-Based Cryptography. In *NuTMiC*. 121–141.

[48] Tim van Erven and Peter Harremoës. 2014. Rényi Divergence and Kullback-Leibler Divergence. *IEEE Trans. Inf. Theory* 60, 7 (2014), 3797–3820.

[49] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *CoRR* abs/1708.07747 (2017).

[50] Andrew Chi-Chih Yao. 1986. How to Generate and Exchange Secrets (Extended Abstract). In *FOCS*. 162–167.

[51] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. 2018. Privacy risk in machine learning: Analyzing the connection to overfitting. In *CSF*. 268–282.

[52] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2021. Understanding Deep Learning (Still) Requires Rethinking Generalization. *Commun. ACM* 64, 3 (2021), 107–115.

[53] Yuqing Zhu and Yu-Xiang Wang. 2019. Poission Subsampled Rényi Differential Privacy. In *ICML*. 7634–7642.