Abstract

Propose-Test-Release (PTR) is a differential privacy framework that works with local sensitivity of functions, instead of their global sensitivity. This framework is typically used for releasing robust statistics such as median or trimmed mean in a differentially private manner. While PTR is a common framework introduced over a decade ago, using it in applications such as robust SGD where we need many adaptive robust queries is challenging. This is mainly due to the lack of Rényi Differential Privacy (RDP) analysis, an essential ingredient underlying the moments accountant approach for differentially private deep learning. In this work, we generalize the standard PTR and derive the first RDP bound for it when the target function has bounded global sensitivity. We show that our RDP bound for PTR yields tighter DP guarantees than the directly analyzed $(\varepsilon, \delta)$-DP. We also derive the algorithm-specific privacy amplification bound of PTR under subsampling. We show that our bound is much tighter than the general upper bound and close to the lower bound. Our RDP bounds enable tighter privacy loss calculation for the composition of many adaptive runs of PTR. As an application of our analysis, we show that PTR and our theoretical results can be used to design differentially private variants for byzantine robust training algorithms that use robust statistics for gradients aggregation. We conduct experiments on the settings of label, feature, and gradient corruption across different datasets and architectures. We show that PTR-based private and robust training algorithm significantly improves the utility compared with the baseline.

1 Introduction

Privacy is a major concern for deploying machine learning (ML). In response, differential privacy (DP) [DMNS06] has become the de-facto measure of privacy. For a differentially private mechanism, the probability distribution of the mechanism’s outputs on a dataset should be close to the distribution of its outputs on the same dataset with any single individual’s data replaced. A general recipe for releasing the value of a function $f$ on dataset $S$ in a differentially private way is adding random noise to $f(S)$ (output perturbation), where noise magnitude should scale with $f$’s global sensitivity.

However, it might be over-conservative to add noise scaled with global sensitivity. There is a line of research on whether we can do better (e.g., [NRS07], [DL09], [LS13], [KNRS13]). Propose-Test-Release (PTR) [DL09] is a framework that improves the general recipe with the notion of local sensitivity.
The main idea of PTR is as follows: instead of adding noise with respect to global sensitivity, we propose an amount of noise that is tolerable for many common queries. When we receive the actual query, we test (in a differentially private way) whether answering the query with the proposed amount of noise is enough for privacy. If the proposed noise is too small for limiting the privacy loss from the actual query, we may refuse to answer the query or respond with a larger noise. PTR works especially well for releasing robust statistics such as median or trimmed mean, as the robust statistics usually have small local sensitivity on most common inputs.

While PTR is a basic framework that ages back to the early days after the introduction of DP, it has not been used for differentially private optimization before. DP-SGD [ACG+16] is the general backbone for differentially private deep learning and optimization. One major challenge of augmenting SGD with PTR and training DP models is the calculation of privacy parameters after a large number of iterations. In this paper, we show that PTR framework can lead to significantly looser privacy bounds. Besides, we also need the bound of privacy amplification by subsampling for PTR, which is the other important support for training DP models.

It allows us to exploit the stochasticity of SGD for the interest of stronger privacy guarantees.

**Technical Overview.** In this work, we derive the Rényi DP bound for PTR, as well as for its Poisson subsampled variant when the target function has bounded global sensitivity. Our bounds make it possible for us to use PTR framework in augmenting private SGD. PTR could be characterized by three mechanisms. The first mechanism is $M_1$ that determines which mechanism to run next. Depending on the outcome of $M_1$ we then either run $M_2$ or $M_2'$. It is often the case that the worst-case privacy loss of one of $M_2$ or $M_2'$ is much larger than the other. Our bound exploits the fact that the worst-case will happen with small probability. Specifically, instead of considering the worst-case privacy loss between $M_2$ and $M_2'$, and naively composing it with $M_1$, we show that for RDP we can tighten the bound by the average privacy loss of $M_2$ and $M_2'$ under the distribution imposed by $M_1$. Direct $(ε, δ)$-DP analysis does not enjoy this benefit; compared with directly analyzing the $(ε, δ)$-DP bound of PTR, we show that first bounding the RDP of PTR and then convert it to $(ε, δ)$-DP can lead to better privacy guarantee. Our proof could serve as a general recipe for analyzing DP/RDP guarantees for composed mechanisms where the privacy loss of each mechanism is adaptively determined. Additionally, we extend our analysis to the RDP of subsampled PTR. Our algorithm-specific analysis (the “white-box bound”) allows us to get tighter privacy amplification bounds, compared with the one obtained by general subsampled RDP bound that supports any mechanisms (the “black-box bound”) [ZW19]. The proof tackled several additional difficulties compared with the analysis for simple Gaussian mechanism [MTZ12]. By numerical verification, we show that our RDP bound for subsampled PTR is much tighter than the black-box bound, and is close to the lower bound (Figure 2).

**Applications of PTR in ML.** PTR is especially suitable for improving the utility of privatizing robust statistics such as trimmed mean, which these functions usually have a much smaller local sensitivity compared to their global sensitivity. A critical application of robust statistics for machine learning is defending against corrupted data. The learning algorithm should be robust in the presence of corrupted data (referred to as Byzantine failure [LSP82]). Since privacy and robustness are two major concerns for ML training, developing techniques that achieve both goals simultaneously is desirable. We demonstrate the application of PTR in incorporating differential privacy with robust SGD methods. A popular technique for robustifying SGD is to replace the mean with some robust statistics (e.g., trimmed mean) for gradient aggregation [YCKB18, AHU+12, GLV21]. We use trimmed mean as an example of showing how to augment robust SGD with DP through PTR. We show that the augmented SGD still maintains the robustness guarantee. We conduct extensive experiments on defending against three kinds of attacks: label, feature, and gradient corruptions, and we show that PTR-based robust SGD achieves much better utility than naively privatizing the robust SGD with global sensitivity.

## 2 Related Work

Since the seminal work of [DL09], the Propose-Test-Release has become a common DP framework mainly used in statistical inference [BAM20, LKO21]. To the best of our knowledge, this work is the first application of PTR in machine learning.
Several prior works (e.g., [LQST12, BBG18, WBK19, ZW19]) focus on the general subsampled DP/RDP bound that supports any mechanisms. [WBK19] derives a general RDP privacy amplification for “sampling without replacement” scheme, and [ZW19] obtains a similar result under Poisson subsampling (i.e., including each data point independently at random with certain probability). Only a small body of recent works study algorithm-specific privacy amplification bound by subsampling. However, most of them focus on subsampled Gaussian mechanism [ACG+16, BDRST18, MFTZ19]. This work derives the first subsampled RDP bound specific for flexible algorithms such as PTR.

Designing machine learning and optimization algorithms that achieve both privacy and Byzantine-robustness is certainly an important direction. Nevertheless, there are only few works on this line so far. [GJP+21] considered the problem of achieving privacy and byzantine resilience in distributed SGD with an untrusted server. The privacy level they considered is essentially local differential privacy, which is orthogonal to our focus. [HKJ20] and [SGA20] aim for both robustness and secure multiparty computation instead of differential privacy. [MZ19] naively applies differential privacy to defend against data poisoning attack. However, they find that DP alone cannot defend adversaries that poison a large fraction of training examples. Aside from the setting of training ML models, [LKKO21] propose a polynomial time algorithm that achieves both goals for mean estimation via privatizing filter-based robust mean estimator [DKK+17]. [EMNZ21] develop a robust and differentially private mean estimator based on exponential mechanism. However, both of their approaches become inefficient (even in polynomial time asymptotically) in high dimensional settings.

3 Background

In this section, we introduce some background on differential privacy, Rényi differential privacy, and privacy-amplification by subsampling. We will also introduce notations as we proceed.

**Differential Privacy.** Differential privacy is a framework for protecting privacy when performing statistical releases on a dataset with sensitive information about individuals (see the surveys [DR+14, Vad17]). Specifically, for a differentially private mechanism, the probability distribution of the mechanism’s outputs of a dataset should be close to the distribution of its outputs on the same dataset with any single individual’s data replaced. To formalize this, we call two datasets $S$, $S'$, each multisets over a data universe $\mathcal{X}$, adjacent if one can be obtained from the other by adding or removing a single element of $\mathcal{X}$. Further, we use $d(S, S')$ to denote the number of times of adding/removing of data points to transform $S$ to $S'$. So $S$ and $S'$ are adjacent if and only if $d(S, S') = 1$.

**Definition 3.1 (Differential Privacy [DMNS06]).** For $\varepsilon, \delta \geq 0$, a randomized algorithm $\mathcal{M} : \text{MultiSets}(\mathcal{X}) \rightarrow \mathcal{Y}$ is $(\varepsilon, \delta)$-differentially private if for every dataset pair $S, S' \in \text{MultiSets}(\mathcal{X})$ such that $d(S, S') = 1$, we have:

$$\forall T \subseteq \mathcal{Y} \quad \Pr[\mathcal{M}(S) \in T] \leq e^\varepsilon \cdot \Pr[\mathcal{M}(S') \in T] + \delta$$

(1)

where the randomness is over the coin flips of $\mathcal{M}$.

**Rényi Differential Privacy (RDP).** Rényi differential privacy (RDP) is a variant of the standard $(\varepsilon, \delta)$-DP that uses Rényi-divergence as a distance metric between the output distributions of $\mathcal{M}(S)$ and $\mathcal{M}(S')$, which is particularly useful in training differentially private machine learning models.

**Definition 3.2 (Rényi Differential Privacy [Mir17]).** We say that a mechanism $\mathcal{M}$ is $(\alpha, \varepsilon)$-RDP with order $\alpha \in (1, \infty)$ if for every dataset pair $S, S' \in \text{MultiSets}(\mathcal{X})$ such that $d(S, S') = 1$, we have:

$$D_\alpha(\mathcal{M}(S)||\mathcal{M}(S')) := \frac{1}{\alpha - 1} \log \mathbb{E}_{\sigma \sim \mathcal{M}(S')} \left[ \left( \frac{\mu_{\mathcal{M}(S)}(\sigma)}{\mu_{\mathcal{M}(S')}(\sigma)} \right)^\alpha \right] \leq \varepsilon$$

(2)

where $\mu_{\mathcal{M}}(\cdot)$ denotes the density function of $\mathcal{M}$’s distribution. Further, we denote the moment

$$E_\alpha(\mathcal{M}(S)||\mathcal{M}(S')) := \mathbb{E}_{\sigma \sim \mathcal{M}(S')} \left[ \left( \frac{\mu_{\mathcal{M}(S)}(\sigma)}{\mu_{\mathcal{M}(S')}^{(\alpha)}} \right)^\alpha \right]$$

and function $f_\alpha(\varepsilon) := \exp((\alpha - 1)\varepsilon)$.

As we can see, $(\alpha, \varepsilon)$-RDP is essentially an upper bound for the moment $E_\alpha(\mathcal{M}(S)||\mathcal{M}(S')) \leq f_\alpha(\varepsilon)$ for all adjacent $S, S'$, where $\varepsilon$ can be viewed as a degree of the privacy loss incurred by running $\mathcal{M}$. A different $\alpha$ typically leads to a different privacy bound $\varepsilon$. Following the convention of literature [ZW19], we view $\varepsilon$ as a function of $\alpha$, and the notation $\varepsilon_\mathcal{M}(\alpha)$ means the algorithm $\mathcal{M}$ obeys
Accountant technique, which composes RDP and then transforms to DP, is a much simpler approach via Gaussian mechanism with a smaller noise than the minimum amount of data points that we need to replace for a local sensitivity bound of the target function, mean) and its robust variant function has an approximation that is a robust statistic of that robust statistics may have a small local sensitivity for most input datasets. The seminal work to single data addition/removal for datasets that are i.i.d. drawn from natural distributions. This means the global sensitivity of the target function may be intolerably large due to some extreme cases.

4.1 Propose-Test-Release

RDP for Gaussian Mechanism:

If the safety margin is large enough, then the algorithm releases \( f_2(S) \) via Gaussian mechanism with a smaller noise \( \sigma_2 \) (usually scaled with \( \tau \)). Otherwise, the algorithm

---

¹ Sometimes the robust statistic itself is the target function, i.e., \( f_1 = f_2 \) in Algorithm.
release $f_1(S)$ with a larger noise $\sigma_1$ (usually scaled with $\mathbb{G}_{f_1}$). The pseudocode is outlined in Algorithm 1.

We make several remarks on Algorithm 1: (1) The traditional version of PTR in the textbooks [DR14, Vad17] simply refuse to output anything when the noisy safety margin $\Delta$ is small. Algorithm 1 is a more general version which output $f_1(S) + \mathcal{N}(0, \sigma_1^2 \mathbf{1}_d)$ when the sensitivity test is failed. The textbook PTR can be thought of as a special case of Algorithm 1 where we set $\sigma_1 \to \infty$. (2) The mechanism used in the Test and Release step can be other mechanisms instead of Laplace and Gaussian. In this paper, we present our results for the PTR instantiated by these two mechanisms since we intend to apply it for differentially private SGD later. Our proof can be easily extended to other mechanisms as well. (3) The threshold $\log(1/(2\delta_0))b$ in Line 3 is chosen so that $\Pr[\text{Lap}(0,b) > \log(1/(2\delta_0))b] = \delta_0$. (4) The global sensitivity of $\Delta(\cdot)$ is 1 for any functions. This could be seen by noticing that, for any pair of adjacent $S, S'$, we have $d(S, \tilde{S}) \leq d(S', \tilde{S}) + 1$ for any dataset $\tilde{S}$.

### Direct DP Analysis of Propose-Test-Release

We show the DP guarantee for the Propose-Test-Release framework. To prove privacy, we need to find the worst pair of adjacent datasets $S$ and $S'$ that incurs the largest privacy loss. It is clear that the worst possible scenario of PTR is when $f_2(S) - f_2(S') > \tau$, while the algorithm still releases $f_2(S)$ with noise scaled with $\tau$. However, note that this worst possible scenario will only happen when both $\mathbb{L}(S)$ and $\mathbb{L}(S')$ are greater than $\tau$. In this case, however, $\Delta$ for both $S$ and $S'$ are 0 since there are no data points we need to change for them to have local sensitivity larger than $\tau$, and thus there is no privacy loss from $\Delta$. Moreover, when $\Delta = 0$, the probability that PTR will release $f_2(S) + \mathcal{N}(0, \sigma_2^2 \mathbf{1}_d)$ is at most $\delta_0$ by construction, which could be simply added to the $\delta$ term in $(\varepsilon, \delta)$-DP. If one of $\mathbb{L}(S)$ and $\mathbb{L}(S')$ is smaller than $\tau$, then we know that $f_2(S) - f_2(S') \leq \tau$, and the overall privacy parameters could be computed by Basic Composition Theorem [DR14]. We follow the above observation for PTR based on the following differential privacy guarantee for PTR.

#### Theorem 4.2 (Direct DP analysis for PTR)

Suppose $\mathbb{G}_{f_1} = \mathbb{G}_{f_2} = 1$ and $\sigma_1 = \sigma_2/\tau$, then Algorithm 1 is $(\varepsilon(b) + \varepsilon(N)^{\delta}(\delta), \delta_0 + \delta)$-DP, where $\varepsilon(b) = e^1/b$ is the DP guarantee for Laplace mechanism, and $\varepsilon(N)^{\delta}(\delta) = \sigma_1 \sqrt{2 \log(1.25/\delta)}$.

### 4.2 RDP of Propose-Test-Release

We then derive the RDP of Propose-Test-Release framework. The major differences between the proofs of $(\varepsilon, \delta)$-DP and RDP bound is that, for $(\varepsilon, \delta)$-DP we can move the probability of running into the bad scenario to the $\delta$ term, while for RDP we need to consider the “average-case” privacy loss. This is in fact an advantage of RDP, which we illustrate it in the following simple example.

#### Comparison between $(\varepsilon, \delta)$-DP and RDP Analysis: A motivating example

Suppose we have two mechanisms $\mathcal{M}_1$ and $\mathcal{M}_2$ who are $(\varepsilon_1, \delta_1)$-DP and $(\varepsilon_2, \delta_2)$-DP, respectively. Consider a simple PTR-like mechanism $\mathcal{M}$ that randomly picks one of mechanisms $\mathcal{M}_1$ and $\mathcal{M}_2$ to execute, each with probability $1 - \delta_0$ and $\delta_0$, respectively. We can only claim that $\mathcal{M}$ is $(\max(\varepsilon_1, \varepsilon_2), (1 - \delta_0)\delta_1 + \delta_0\delta_2)$-DP, or if we know $\varepsilon_2 > \varepsilon_1$ we can also move the “bad case” probability $\delta_0$ to the $\delta$ term and obtain $(\varepsilon_1, \delta_1 + \delta)$-DP. However, if we know the RDP guarantee of $\mathcal{M}_1$ and $\mathcal{M}_2$ as $E_{\alpha}(\mathcal{M}_1(S) \| \mathcal{M}_1(S')) \leq f_\alpha(\varepsilon_1)$ and $E_{\alpha}(\mathcal{M}_2(S) \| \mathcal{M}_2(S')) \leq f_\alpha(\varepsilon_2)$ then $E_{\alpha}(\mathcal{M}(S) \| \mathcal{M}(S'))$.

---

[1] Details of derivation can be found in Appendix A.
[2] For the actual PTR, the $\delta_0$ is not fixed but depends on the input dataset.
[3] Recall that $f_\alpha(\varepsilon) = \exp((\alpha - 1)\varepsilon)$ where if $\mathcal{M}$ is $(\alpha, \varepsilon)$-RDP then $E_{\alpha}(\mathcal{M}(S) \| \mathcal{M}(S')) \leq f_\alpha(\varepsilon)$. 

---

**Algorithm 1: Propose-Test-Release with Laplace and Gaussian mechanisms.**

**input**: $S$ – dataset, $f_1$ – target function, $f_2$ – robust statistic, $\tau$ – proposed local sensitivity bound of $f_2$, $\sigma_1, \sigma_2, b$ – Gaussian/Laplace noise scales ($\sigma_1 > \sigma_2$), $\delta_0$ – failure probability.

1. $\Delta \leftarrow \min_{\tilde{S} \in \hat{S} \in \mathbb{L}(f_1(\tilde{S}) > \tau)} d(S, \tilde{S})$.
2. $\tilde{\Delta} \leftarrow \Delta + \text{Lap}(0,b)$.
3. if $\tilde{\Delta} \leq \log(1/(2\delta_0))b$ then
   4. return $f_1(S) + \mathcal{N}(0, \sigma_1^2 \mathbf{1}_d)$
5. else
   6. return $f_2(S) + \mathcal{N}(0, \sigma_2^2 \mathbf{1}_d)$

---

5.
We further derive the RDP for Poisson subsampled PTR. Here, “subsampled” means the input dataset will be subsampled first before feeding to PTR. Poisson subsampling means each data variable for PTR’s output on dataset $S$ will be subsampled first before feeding to PTR. Poisson subsampling means each data point $x$ will be included with probability $q$ independently. One way of obtaining the RDP for any subsampled mechanism is by simply plugging in the RDP bound of the original algorithm into the privacy amplification formula for general mechanisms (e.g., [WBK19, ZW19]). Here, we directly derive the RDP for Poisson subsampled PTR in a white-box manner. Denote $M_0$ as the random variable for PTR’s output on dataset $S$, and $M_1$ for PTR’s output on dataset $S’ = S \cup \{x\}$, and $M = (1 - q)M_0 + qM_1$. Compared with the analysis for subsampled Gaussian mechanism by [MZ19], there are two main difficulties for extending the arguments of quasi-convexity of Rényi divergence for PTR: (1) the distribution of PTR’s output may not be centrally symmetric, thus we need to bound both $D_2(M_0||M_0)$ and $D_2(M_0||M)$. (2) the conditional distribution $M|\Delta$ cannot be decomposed as $(1 - q)M_0|\Delta + qM_1|\Delta$. A big part of our novelty in the proof is about addressing these two challenges. We defer details of the proof to Appendix A.

Figure 1: The $\varepsilon$ parameter of the $(\varepsilon, \delta)$-DP guarantee of PTR when $\delta = 10^{-5}$ for different noise scales. We convert the RDP bound in Theorem 4.3 to $(\varepsilon, \delta)$-DP by the RDP-DP conversion formula from [BBG+20], and compare it with the $\varepsilon$ obtained from the direct analysis in Theorem 4.2. For the bound converted from RDP, we search for the optimal $\alpha \in [1, 200]$. The bound is constant across different $\delta_0$ since when $\delta_0$ is small, the RDP for PTR will take the second term in (3).
An exciting byproduct during our research on subsampled PTR is the privacy amplification bound which leads to a better
we can see from Figure 2, Theorem 4.4 (orange curve) is much tighter than the black-box bound (blue curve). Moreover, it is very close to the lower bound of privacy amplification by 

This bound may not be very interpretable, which is a typical feature for the privacy amplification bounds as they are meant to be implemented in practice. After all, constant matters for differential privacy practitioners! To show the tightness of our bound, we plug in the RDP of original PTR (Theorem 4.3) to the current tightest privacy amplification formula for general mechanisms derived in [ZW19] (the “black-box bound”), and compare it with our white-box bound in Theorem 4.4. As we can see from Figure 2, Theorem 4.4 (orange curve) is much tighter than the black-box bound (blue curve). Moreover, it is very close to the lower bound of privacy amplification by [ZW19] (green line), which means that Theorem 4.4 is near optimal. In Figure 3, we illustrate the application of subsampled RDP bound in Moment Accountant. We can see that the privacy parameters for the composed mechanism obtained based on our white-box bound (Theorem 4.4) is tighter than the one by black-box bound, as well as the one by directly composing Theorem 4.2 with strong composition theorem [KOV15]. We remark that Direct Analysis achieves lower privacy loss with very few iterations since we set \( \delta_0 = 10^{-8} \) to allow more iterations for Strong Composition of \((\varepsilon, \delta)\)-DP, which leads to a better \( \varepsilon \) for a single iteration (see Figure 4).

An exciting byproduct during our research on subsampled PTR is the privacy amplification bound for a \( \mathcal{M} \) that outputs \( \mathcal{M}_1(S), \mathcal{M}_2(S) \) sequentially and independently. This result is important since such a \( \mathcal{M} \) is instantiated in many existing techniques in improving differentially private deep learning. We defer this result to the Appendix.

## 5 Differentially Private and Robust SGD with Propose-Test-Release

In this section, we demonstrate the application of the RDP for (subsampled) PTR algorithm in privatizing robust SGD algorithms.

**Attack Model.** Mini-batch SGD is a common method for training deep neural networks. Despite its strong convergence properties in the standard settings, it is well known that even a small fraction of corrupted gradients can lead SGD to an arbitrarily poor solution [BTN09, BBC11]. An important attack model is called Byzantine contamination framework or Byzantine failure [LSP82]. Consider an
optimization problem with \( n \) stochastic gradient oracles; at each iteration there are up to \( F \) gradient oracles are corrupted (usually referred as Byzantine agents in the context of distributed SGD). The identity of corrupted oracles is a priori unknown. As the corrupted gradients can be arbitrarily skewed, this attack model is able to capture many important scenarios including corruption in feature (e.g., existence of outliers), corruption in gradients (e.g., hardware failure, unreliable communication channels during distributed training) and corruption in labels (e.g. label flip (backdoor) attacks).

A popular class of defense strategies against Byzantine failure is to use robust gradient aggregation rules such as trimmed-mean GLV21, coordinate-wise median YCKB18, and geometric median AHJ21. Since both privacy and robustness are essential for ML applications, it is important to develop learning algorithms that achieve differential privacy and robustness simultaneously.

**Baseline: privatizing robust SGD with Gaussian Mechanism.** A natural (but naive) way to accomplish such a goal for SGD-based algorithms is to privatize each iteration of robust SGD through Gaussian mechanism. Specifically, in each iteration, we add Gaussian noise to the aggregated gradient, where noise magnitude scale with the global sensitivity of the robust aggregation function. The privacy parameter for the entire process is computed through Moment Accountant. The global sensitivity can be obtained by either upper bounding the Lipschitz constant of the objective function, or clipping each gradient according to a certain threshold. However, this method could potentially add over-conservative noise to the aggregated gradients, as the local sensitivity of robust aggregation functions is usually very low for most input gradient sets.

**Privatizing robust SGD with PTR.** We improve the naive method with the PTR framework, where we instantiate it by showing how to privatize trimmed-mean SGD method from GLV21. We first define trimmed-sum aggregation function, and show that the \( \Delta \) for trimmed-sum could be computed efficiently. Given a dataset \( S = \{x_1, \ldots, x_m\} \), let \( x_{(k)} \) denote the \( k \)th smallest data point among \( S \) in \( \ell_2 \) norm, i.e., \( \|x_{(1)}\| \leq \|x_{(2)}\| \leq \cdots \leq \|x_{(m)}\| \). The \( F \)-trimmed sum of \( S \) is defined as \( \text{TSUM}_F(S) = \sum_{i=1}^{m-F} x_{(i)} \) if \( m > F \), or simply 0 if \( m \leq F \). Define \( \mathbb{LS}_f^{(r)}(S) = \sup_{S,F,\text{d}(S,F)=r} \mathbb{LS}_f(S) \), which is the maximum local sensitivity that can be achieved by adding/removing \( r \) elements from \( S \). Thus, \( \mathbb{LS}_f^{(0)}(S) = \mathbb{LS}_f(S) \), and the safety margin \( \Delta(S) = \min\{r : \mathbb{LS}_f^{(r)}(S) > \tau\} \). Therefore, as long as we can efficiently compute \( \mathbb{LS}_f^{(r)}(S) \), we can efficiently compute \( \Delta(S) \) in linear time by simply enumerate all \( r \) from 0 to \( m \) and terminate once \( \mathbb{LS}_f^{(r)}(S) > \tau \). In fact, \( \mathbb{LS}_f^{(r)}(S) \) can be computed in \( O(1) \) time.

**Theorem 5.1.** \( \mathbb{LS}_f^{(r)}(\text{TSUM}_p(S)) = \|x_{(m-F+1+r)}\| \) if \( r \leq F - 1 \), or \( G \mathbb{LS}_f^{(r)}(\text{TSUM}_p(S)) \) if \( r > F - 1 \).

Back to the goal of designing differentially private and robust SGD. Fix a positive integer \( F \) which is a potential upper bound for the number of corrupted gradients. We instantiate a PTR-based gradient aggregation algorithm by instantiating \( f_1(\cdot) \) as \( \text{SUM}(S) = \sum_{x \in S} x \), and \( f_2(\cdot) \) as \( \text{TSUM}_F(\cdot) \). We simply plugging in this PTR algorithm as the gradient aggregation function for regular SGD. We show

---

\(^3\)We use trimmed-sum instead of trimmed-mean for the ease of analysis, as trimmed-mean essentially just scales the learning rate.
We make two remarks of our specific design choices for implementing PTR-based SGD in practice. First, we increase the value of $F$ for improving model performance. Specifically, we compare the utility-privacy tradeoff between the algorithm that privatize trimmed mean-based SGD with PTR (TSGD + PTR), and the baseline algorithm that privatize trimmed mean-based SGD with Gaussian mechanism (TSGD + Gaussian).

**Theorem 5.2 (Informal).** When the loss function is Lipschitz in model parameters, SGD with PTR instantiated by SUM and TSUM provides convergence guarantee for update to $F$ gradients being corrupted arbitrarily for any $F < m/2$.

We make two remarks of our specific design choices for implementing PTR-based SGD in practice. First, we increase the value of $F$ for improving model performance. Specifically, we compare the utility-privacy tradeoff between the algorithm that privatize trimmed mean-based SGD with PTR (TSGD + PTR), and the baseline algorithm that privatize trimmed mean-based SGD with Gaussian mechanism (TSGD + Gaussian).

**Experiment Settings.** We evaluate the utility-privacy tradeoff of the two robust and differentially private SGD algorithms under three common types of attack: label, feature, and communicated gradient corruption. For label corruption, we randomly flip of label of certain amount of data points. For feature corruption, we add Gaussian noise from $\mathcal{N}(0, 100)$ directly to the corrupted images. For

---

Table 1: Model Accuracy under different privacy budgets and corruption settings. Every statistic is averaged over 5 runs with different random seed. The improvement of TSGD + PTR over TSGD + Gaussian is highlighted in the red text. $\epsilon$s are chosen differently for different datasets since the best accuracy-privacy tradeoff point varied for datasets. ‘CR’ means corruption ratio.

The convergence guarantee of this algorithm in the presence of at most $F$ gradients being corrupted for the case that the loss function is Lipschitz.

### Table 1: Model Accuracy under different privacy budgets and corruption settings.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Corruption Type</th>
<th>CR</th>
<th>$\epsilon = 3.0$</th>
<th>$\epsilon = 5.0$</th>
<th>$\epsilon = 7.0$</th>
<th>$\epsilon = 10.0$</th>
<th>$\epsilon = 2.0$</th>
<th>$\epsilon = 2.5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>Label</td>
<td>0</td>
<td>87.53%</td>
<td>91.43% (+3.9%)</td>
<td>81.31%</td>
<td>92.8% (+11.4%)</td>
<td>57.94%</td>
<td>58.92% (+0.98%)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.1</td>
<td>90.15%</td>
<td>93.29% (+3.13%)</td>
<td>89.15%</td>
<td>94.60% (+5.45%)</td>
<td>55.48%</td>
<td>56.26% (+0.78%)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.2</td>
<td>91.33%</td>
<td>92.704% (+1.374%)</td>
<td>92.47%</td>
<td>94.08% (+1.614%)</td>
<td>55.7%</td>
<td>56.628% (+0.93%)</td>
</tr>
<tr>
<td>Feature</td>
<td></td>
<td>0</td>
<td>89.47%</td>
<td>92.28% (+2.812%)</td>
<td>86.53%</td>
<td>93.38% (+6.854%)</td>
<td>55.13%</td>
<td>55.526% (+0.398%)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.1</td>
<td>91.71%</td>
<td>92.29% (+0.582%)</td>
<td>86.99%</td>
<td>92.86% (+5.872%)</td>
<td>55.81%</td>
<td>57.29% (+1.48%)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.2</td>
<td>90.27%</td>
<td>91.7% (+1.43%)</td>
<td>88.41%</td>
<td>91.68% (+3.27%)</td>
<td>54.45%</td>
<td>56.31% (+1.86%)</td>
</tr>
<tr>
<td>Gradient</td>
<td></td>
<td>0</td>
<td>91.13%</td>
<td>91.45% (+0.32%)</td>
<td>88.70%</td>
<td>90.31% (+1.61%)</td>
<td>57.94%</td>
<td>58.92% (+0.98%)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.1</td>
<td>91.45%</td>
<td>91.45% (+0.32%)</td>
<td>57.94%</td>
<td>58.92% (+0.98%)</td>
<td>57.21%</td>
<td>58.96% (+1.75%)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.2</td>
<td>91.45%</td>
<td>91.45% (+0.32%)</td>
<td>57.21%</td>
<td>58.96% (+1.75%)</td>
<td>56.11%</td>
<td>57.48% (+1.37%)</td>
</tr>
</tbody>
</table>

The convergence guarantee of this algorithm in the presence of at most $F$ gradients being corrupted for the case that the loss function is Lipschitz.

**Theorem 5.2 (Informal).** When the loss function is Lipschitz in model parameters, SGD with PTR instantiated by SUM and TSUM provides convergence guarantee for update to $F$ gradients being corrupted arbitrarily for any $F < m/2$. We make two remarks of our specific design choices for implementing PTR-based SGD in practice. First, we increase the value of $F$ for improving model performance. Specifically, we compare the utility-privacy tradeoff between the algorithm that privatize trimmed mean-based SGD with PTR (TSGD + PTR), and the baseline algorithm that privatize trimmed mean-based SGD with Gaussian mechanism (TSGD + Gaussian).

**Experiment Settings.** We evaluate the utility-privacy tradeoff of the two robust and differentially private SGD algorithms under three common types of attack: label, feature, and communicated gradient corruption. For label corruption, we randomly flip of label of certain amount of data points. For feature corruption, we add Gaussian noise from $\mathcal{N}(0, 100)$ directly to the corrupted images. For

---

6 This does not introduce extra privacy leakage as DP is closed under post-processing.
gradient corruption, we add Gaussian noise from $\mathcal{N}(0, 100)$ to the true gradients. We experiment on three classic datasets, MNIST, CIFAR10, and EMNIST. We vary different corruption ratios and privacy parameter $\varepsilon$. Experiment details are deferred to Appendix.

**Results.** The comparison of model test accuracy under given corruption settings and privacy parameters are summarized in Table [1]. We highlight the improvement of TSGD + PTR over the baseline TSGD + Gaussian in red texts. As we can see, for all settings, TSGD + PTR outperforms and often works significantly better than TSGD + Gaussian. This demonstrates that, while TSGD + PTR may introduce extra privacy loss in the test step, the performance gain from adding smaller noise in the release step overshadows it.

We also observe two interesting phenomena in the experiment: (1) a higher corruption ratio may not necessarily lead to worse model performance for trimmed mean-based robust SGD, especially for MNIST dataset. This is because the high norm gradients can be either corrupted, or benign but the partially-trained model misclassifies the corresponding data points. The latter case is extremely important for improving model performance compared with the gradients of data points are already being classified correctly. When the corruption ratio is small, more benign gradients are being trimmed, which may lead to worse model performance. (2) For TSGD + Gaussian, a larger privacy budget may not lead to better model performance on MNIST dataset. This is because when the training accuracy reaches the peak, there are many trimmed gradients whose corresponding training data points are already correctly classified; continuing training without those data points and with large noise may result in catastrophic forgetting.

### 6 Conclusion and Limitation

This work derives the Rényi Differential Privacy for propose-test-release framework as well as its subsampled version. With the RDP bound for the PTR framework, this work demonstrate the application of PTR in training differentially private and robust models. One limitation of PTR is that it does not work well in privatizing coordinate-wise median in high-dimensional space. The global sensitivity of coordinate-wise median is far greater than the one for mean, which results in huge privacy loss.

**Acknowledgement**

We are grateful to anonymous reviewers at NeurIPS for valuable feedback. This work was supported in part by the National Science Foundation under grants CNS2131910, CNS-1553437, CNS-1553301, CNS-1704105, and CNS-1953786, the ARL’s Army Artificial Intelligence Innovation Institute (A2I2), the Office of Naval Research Young Investigator Award, the Army Research Office Young Investigator Prize, Schmidt DataX award, Princeton E-ffiliates Award, Amazon-Virginia Tech Initiative in Efficient and Robust Machine Learning, and Princeton’s Gordon Y. S. Wu Fellowship.
References


Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope? [Yes] See Section 4 and 5, in particular Figure 1, 2 and Table 1.
   (b) Did you describe the limitations of your work? [Yes] See Section 6 for a discussion on limitations.
   (c) Did you discuss any potential negative societal impacts of your work? [No] We do not see any potential negative societal impacts of our work.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Appendix A.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix A.

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We provide our code in supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix B for full details.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We report error bars for Table 1 in Appendix B.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix B for full details.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We cite each dataset and model architecture we use, see full details in Appendix B.
   (b) Did you mention the license of the assets? [Yes] See Appendix B.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you’re using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]