DP-SGD with weight clipping

Antoine Barczewski, Jan Ramon

DP-SGD with weight clipping

Antoine Barczewski[1] and Jan Ramon[2]

[1]Université Lille, Inria
[2]Inria

June 17, 2024

Abstract

Recently, due to the popularity of deep neural networks and other methods whose training typically relies on the optimization of an objective function, and due to concerns for data privacy, there is a lot of interest in differentially private gradient descent methods. To achieve differential privacy guarantees with a minimum amount of noise, it is important to be able to bound precisely the sensitivity of the information which the participants will observe. In this study, we present a novel approach that mitigates the bias arising from traditional gradient clipping. By leveraging a public upper bound of the Lipschitz value of the current model and its current location within the search domain, we can achieve refined noise level adjustments. We present a new algorithm with improved differential privacy guarantees and a systematic empirical evaluation, showing that our new approach outperforms existing approaches also in practice.

Keywords: Machine Learning, Differential Privacy, Optimization.

1 Introduction

While machine learning allows for extracting statistical information from data with both high economical and societal value, there is a growing awareness of the risks for data privacy and confidentiality. Differential privacy Dwork and Roth (2013) has emerged as an important metric for studying statistical privacy.

Due to the popularity of deep neural networks (DNNs) and similar models, one of the recently most trending algorithmic techniques in machine learning has been stochastic gradient descent (SGD), which is a technique allowing for iteratively improving a candidate model using the gradient of the objective function on the data.

A popular class of algorithms to realize differential privacy while performing SGD is the DP-SGD algorithm Abadi et al. (2016) and its variants. Essentially, these algorithms iteratively compute gradients, add differential privacy noise, and use the noisy gradient to update the model. To determine the level of differential privacy achieved, one uses an appropriate composition rule to bound the total information leaked in the several iterations.

To achieve differential privacy with a minimum amount of noise, it is important to be able to bound precisely the sensitivity of the information which the participants will observe. One approach is to bound the sensitivity of the gradient by assuming the objective function is Lipschitz continuous Bassily et al. (2014). Various improvements exist in the case one can make additional assumptions about the objective function. For example, if the objective function is strongly convex, one can bound the number of iterations needed and in that way avoid to have to distribute the available privacy budget over too many iterations Bassily et al. (2019). In the case of DNN, the objective function is not convex and typically not even Lipschitz continuous. Therefore, a common method is to 'clip' contributed

gradients Abadi et al. (2016), i.e., to divide gradients by the maximum possible norm they may get. These normalized gradients have bounded norm and hence bounded sensitivity.

In this paper, we argue that gradient clipping may not lead to optimal statistical results (see Section 4), and we propose instead to use weight clipping, an idea suggested in Ziller et al. (2021) but to the best of our knowledge not investigated yet in depth. Moreover, we also propose to consider the maximum gradient norm given the current position in the search space rather than the global maximum gradient norm, as this leads to additional advantages. In particular, our contributions are as follows:

- We introduce an novel approach, applicable to any feed-forward neural network, to compute gradient sensitivity that when applied in DP-SGD eliminates the need for gradient clipping. This strategy bridges the gap between Lipschitz-constrained neural networks and DP.
- We present a new algorithm, Lip-DP-SGD, that enforces bounded sensitivity of the gradients We argue that our approach, based on weight clipping, doesn't suffer from the bias which the classic gradient clipping can cause.
- We present an empirical evaluation, confirming that on a range of popular datasets our proposed method outperforms existing ones.

The remainder of this paper is organized as follows. First, we review a number of basic concepts, definitions and notations in Section 2. Next, we present our new method in Section 3 and present an empirical evaluation in Section 4. We discuss related work in Section 5. Finally, we provide conclusions and directions for future work in Section 6.

2 Preliminaries and background

In this section, we briefly review differential privacy, empirical risk minimization (ERM) and differentially private stochastic gradient descent (DP-SGD).

We will denote the space of all possible instances by $\mathcal{Z}$ and the space of all possible datasets by $\mathcal{Z}^*$. We will denote by $[N] = \{1 \dots N\}$ the set of the N smallest positive integers.

2.1 Differential Privacy

An algorithm is differentially private if even an adversary who knows all but one instances of a dataset can't distinguish from the output of the algorithm the last instance in the dataset. More formally:

Definition 1 (adjacent datasets). *We say two datasets $Z_1, Z_2 \in \mathcal{Z}^*$ are adjacent, denoted $Z_1 \sim Z_2$, if they differ in at most one element. We denote by $\mathcal{Z}^*_\sim$ the space of all pairs of adjacent datasets.*

Definition 2 (differential privacy Dwork and Roth (2013)). *Let $\epsilon > 0$ and $\delta > 0$. Let $\mathcal{A} : \mathcal{Z}^* \to \mathcal{O}$ be a randomized algorithm taking as input datasets from $\mathcal{Z}^*$. The algorithm $\mathcal{A}$ is (ϵ, δ)-differentially private ((ϵ, δ)-DP) if for every pair of adjacent datasets $(Z_1, Z_2) \in \mathcal{Z}^*_\sim$, and for every subset $S \subseteq \mathcal{O}$ of possible outputs of $\mathcal{A}$, $P(\mathcal{A}(Z_1) \subseteq S) \leq e^\epsilon P(\mathcal{A}(Z_2) \subseteq S) + \delta$. If $\delta = 0$ we also say that $\mathcal{A}$ is ϵ-DP.*

If the output of an algorithm $\mathcal{A}$ is a real number or a vector, it can be privately released thanks to differential privacy mechanisms such as the Laplace mechanism or the Gaussian mechanism Dwork et al. (2006). While our ideas are more generally applicable, in this paper we will focus on the Gaussian mechanism as it leads to simplier derivations. In particular, the Gaussian mechanism adds Gaussian noise to a number or vector which depends on its sensitivity on the input.

Definition 3 (sensitivity). *The ℓ_2-sensitivity of a function $f : \mathcal{Z} \to \mathbb{R}^p$ is*

$$s_2(f) = \max_{Z_1, Z_2 \in \mathcal{Z}^*_\sim} \|f(Z_1) - f(Z_2)\|_2$$

Lemma 4 (Gaussian mechanism). *Let $f : \mathcal{Z} \to \mathbb{R}^p$ be a function. The Gaussian mechanism transforms f into $\hat{f}$ with $\hat{f}(Z) = f(Z) + b$ where $b \sim \mathcal{N}(0, \sigma^2 I_p) \in \mathbb{R}^p$ is Gaussian distributed noise. If the variance satisfies $\sigma^2 \geq 2\ln(1.25/\delta)(s_2(f))^2/\epsilon^2$, then $\hat{f}$ is (ϵ, δ)-DP.*

2.2 Empirical risk minimization

Unless made explicit otherwise we will consider databases $Z = \{\mathrm{z}_i\}_{i=1}^n$ containing n instances $\mathrm{z}_i = (\mathrm{x}_i, \mathrm{y}_i) \in \mathcal{X} \times \mathcal{Y}$ with $\mathcal{X} = \mathbb{R}^p$ and $\mathcal{Y} = \{0, 1\}$ sampled identically and independently (i.i.d.) from an unknown distribution on $\mathcal{Z}$. We are trying to build a model $f_\theta : \mathcal{X} \to \hat{\mathcal{Y}}$ (with $\hat{\mathcal{Y}} \subseteq \mathbb{R}$) parameterized by $\theta \in \Theta \subseteq \mathbb{R}^p$, so it minimizes the expected loss $\mathcal{L}(\theta) = \mathbb{E}_z[\mathcal{L}(\theta; \mathrm{z})]$, where $\mathcal{L}(\theta; \mathrm{z}) = \ell(f_\theta(\mathrm{x}), \mathrm{y})$ is the loss of the model f_θ on data point z. One can approximate $\mathcal{L}(\theta)$ by

$$\hat{R}(\theta; Z) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(\theta; \mathrm{z}_i) = \frac{1}{n} \sum_{i=1}^{n} \ell(f_\theta(\mathrm{x}_i), \mathrm{y}_i),$$

the empirical risk of model f_θ. Empirical Risk Minimization (ERM) then minimizes an objective function $F(\theta, Z)$ which adds to this empirical risk a regularization term $\psi(\theta)$ to find an estimate $\hat{\theta}$ of the model parameters:

$$\hat{\theta} \in \arg\min_{\theta \in \Theta} F(\theta; Z) := \hat{R}(\theta; Z) + \gamma\psi(\theta)$$

where $\gamma \geq 0$ is a trade-off hyperparameter.

Feed forward neural networks An important and easy to analyze class of neural networks are the feed forward networks (FNN). A FNN is a direct acyclic graph where connections between nodes don't form cycles.

Definition 5. *A FNN $f_\theta : \mathbb{R}^n \to \mathbb{R}^m$ is a function which can be expressed as*

$$f_\theta = f^{(K)}_{\theta_K} \circ \ldots \circ f^{(1)}_{\theta_1}$$

where $f^{(k)}_{\theta_k} : \mathbb{R}^{n_k} \to \mathbb{R}^{n_{k+1}}$. $f^{(k)}_{\theta_k}$ is the k-th layer function parameterized by θ_k for $1 \leq k \leq K$. We denote the input of $f^{(k)}_{\theta_k}$ by x_k and its output by x_{k+1}. Here, $\theta = (\theta_1 \ldots \theta_K)$, $n = n_1$ and $m = n_{K+1}$.

Common layers include fully connected layers, convolutional layers and activation layers. Parameters of the first two correspond to weight and bias matrices, $\theta_k = (W_k, B_k)$, while activation layers have no parameter, $\theta_k = ()$.

2.3 Stochastic gradient descent

To minimize $F(\theta, Z)$, one can use gradient descent, i.e., iteratively for a number of time steps $t = 1 \ldots T$ one computes a gradient $g^{(t)} = \nabla F(\tilde{\theta}^{(t)}, Z)$ on the current model $\tilde{\theta}^{(t)}$ and updates the model setting $\tilde{\theta}^{(t+1)} = \tilde{\theta}^{(t)} - \eta(t) g^{(t)}$ where $\eta(t)$ is a learning rate. Stochastic gradient descent (SGD) introduces some randomness and avoids the need to recompute all gradients in each iteration by sampling in each iteration a batch $V \subseteq Z$ and computing an approximate gradient $\hat{g}_t = \frac{1}{|V|} \left(\sum_{i=1}^{|V|} \nabla \mathcal{L}(\tilde{\theta}^{(t)}, v_i) + b^{(t)} \right) + \gamma \nabla \psi(\theta)$.

To avoid leaking sensitive information, Abadi et al. (2016) proposes to add noise to the gradients. Determining good values for the scale of this noise has been the topic of several studies. One simple strategy starts by assuming an upper bound for the norm of the gradient. Let us first define Lipschitz functions:

Definition 6 (Lipschitz function). *Let $L^g > 0$. A function f is L^g-Lipschitz with respect to some norm $\|\cdot\|$ if for all $\theta, \theta' \in \Theta$ there holds $\|f(\theta) - f(\theta')\| \leq L^g \|\theta - \theta'\|$. If f is differentiable and $\|\cdot\| = \|\cdot\|_2$, the above property is equivalent to:*

$$\|\nabla f(\theta)\|_2 \leq L^g, \quad \forall \theta \in \Theta$$

We call the smallest value L^g for which f is L^g-Lipschitz the Lipschitz value of f.

Then, from the model one can derive a constant L^g such that the objective function is L^g-Lipschitz, while knowing bounds on the data next allows for computing a bound on the sensitivity of the gradient. Once one knows the sensitivity, one can determine the noise to be added from the privacy parameters as in Lemma 4. The classic DP-SGD algorithm Abadi et al. (2016), which we recall in Algorithm 3 in Appendix A for completeness, clips the gradient of each instance to a maximum value C (i.e., scales down the gradient if its norm is above C) and then adds noise based on this maximal norm C.

$$\tilde{g}_t = \frac{1}{|V|}\left(\sum_{i=1}^{|V|} \text{clip}_C\left(\nabla_{\tilde{\theta}} \mathcal{L}\left(\tilde{\theta}^{(t)}, v_i\right)\right) + b_t\right) + \gamma \nabla \psi(\theta)$$

where b_t is appropriate noise and where

$$\text{clip}_C(v) = v.\min\left(1, \frac{C}{\|v\|}\right).$$

2.4 Regularization

Several papers Ioffe and Szegedy (2015); Wu and He (2018) have pointed out that regularization can help to improve the performance of stochastic gradient descent. Although batch normalization Ioffe and Szegedy (2015) does not provide protection against privacy leakage, group normalization Wu and He (2018) has the potential to do so De et al. (2022). De et al. (2022) combines group normalization with DP-SGD, the algorithm to which we propose an improvement in the current paper. Group normalization is a technique adding specific layers, called group normalization layers, to the network. Making abstraction of some elements specific to image datasets, we can formalize it as follows.

For a vector v, we will denote the dimension of v by $|v|$, i.e., $v \in \mathbb{R}^v$.

If the k-th layer is a normalization layer, then there holds $|x_k| = |x_{k+1}|$. Moreover, the structure of the normalization layer defines a partitioning $\Gamma_k = \{\Gamma_{k,1} \ldots \Gamma_{k,|G|}\}$ of $[|x_k|]$, i.e., a partitioning of the components of x_k. The components of x_k and x_{k+1} are then grouped, and we define $x_k^{(k:q)} = (x_{k,j})_{j \in \Gamma_{k,q}}$, i.e., $x_k^{(k:q)}$ is a subvector containing a group of components. Similarly, $x_{k+1}^{(k:q)} = (x_{k+1,j})_{j \in \Gamma_{k,q}}$. Then, the k-th layer performs the following operation:

$$x_{k+1}^{(k:q)} = f_{\theta_k}^{(k)}(x_k^{(k:q)}) = \frac{1}{\sigma^{(k:q)}}\left(x_k^{(k:q)} - \mu^{(k:q)}\right), \tag{1}$$

(but note we will adapt this in Eq (5)) where

$$\begin{aligned}
\mu^{(k:q)} &= \frac{1}{|\Gamma_{k,q}|}\sum_{j=1}^{|\Gamma_{k,q}|} x_{k,j}, \\
\sigma^{(k:q)} &= \left(\frac{1}{|\Gamma_{k,q}|}\sum_{j=1}^{|\Gamma_{k,q}|}\left(x_{k,j} - \mu^{(k:q)}\right)^2 + \kappa\right)^{1/2},
\end{aligned}$$

with κ a small constant.

Various feature normalization methods primarily vary in their definitions of the partition of features $\Gamma_{k,q}$.

3 Our approach

In this work, we constrain the objective function to be Lipschitz, and exploit this to determine sensitivity. An important advantage is that while traditional DP-SGD controls sensitivity via gradient clipping of each sample separately, our new method estimates gradient sensitivity based on only the model in a data-independent way. This is grounded in Lipschitz-constrained model literature (Section 5), highlighting the connection between the Lipschitz value for input and parameter. Subsection 3.1 demonstrates the use of backpropagation for gradient sensitivity estimation. Subsection 3.2 delves into determining an upper Lipschitz bound, and in 3.3, we introduce Lip-DP-SGD, a novel algorithm ensuring privacy without gradient clipping.

3.1 Backpropagation

Consider a feed-forward network f_θ. We define $\mathcal{L}_k(\theta,(x_k,y)) = \ell\left(\left(f_{\theta_K}^{(K)} \circ \ldots \circ f_{\theta_k}^{(k)}\right)(x_k), y\right)$. For feed-forward networks, backpropagation relies on the subsequent recursive equations:

$$\begin{aligned}\frac{\partial \mathcal{L}_k}{\partial x_k} &= \frac{\partial \mathcal{L}_{k+1}}{\partial x_{k+1}}\frac{\partial x_{k+1}}{\partial x_k} = \frac{\partial \mathcal{L}_{k+1}}{\partial x_{k+1}}\frac{\partial f_{\theta_k}^{(k)}}{\partial x_k} \\ \frac{\partial \mathcal{L}_k}{\partial \theta_k} &= \frac{\partial \mathcal{L}_{k+1}}{\partial x_{k+1}}\frac{\partial x_{k+1}}{\partial \theta_k} = \frac{\partial \mathcal{L}_{k+1}}{\partial x_{k+1}}\frac{\partial f_{\theta_k}^{(k)}}{\partial \theta_k}.\end{aligned} \tag{2}$$

Note that θ_k and x_k are vectors, so also $\frac{\partial \mathcal{L}_k}{\partial x_k}$, $\frac{\partial \mathcal{L}_k}{\partial \theta_k}$ and $\frac{\partial \mathcal{L}_{k+1}}{\partial x_{k+1}}$ are vectors, and $\frac{\partial f_{\theta_k}^{(k)}}{\partial x_k}$ and $\frac{\partial f_{\theta_k}^{(k)}}{\partial \theta_k}$ are Jacobian matrices. In terms of 2-norms there holds

$$\begin{aligned}\left\|\frac{\partial \mathcal{L}_k}{\partial x_k}\right\|_2 &\leq \left\|\frac{\partial \mathcal{L}_{k+1}}{\partial x_{k+1}}\right\|_2 \left\|\frac{\partial f_{\theta_k}^{(k)}}{\partial x_k}\right\|_2 \\ \left\|\frac{\partial \mathcal{L}_k}{\partial \theta_k}\right\|_2 &\leq \left\|\frac{\partial \mathcal{L}_{k+1}}{\partial x_{k+1}}\right\|_2 \left\|\frac{\partial f_{\theta_k}^{(k)}}{\partial \theta_k}\right\|_2\end{aligned} \tag{3}$$

We will use l_k to denote an upper bound of $\max_{x_k,y} \|\frac{\partial \mathcal{L}_k(\theta,x_k,y)}{\partial x_k}\|_2$ and Δ_k to denote the upper bound of $\max_{x_k} \|\frac{\partial \mathcal{L}_k}{\partial \theta_k}\|_2$. In particular, we will ensure that $l_{K+1} \geq \max_{x_{K+1},y} \|\frac{\partial \ell}{x_{K+1}}(x_{K+1},y)\|_2$ and

$$\begin{aligned} l_k &\leq l_{k+1} \max_{x_k} \left\|\frac{\partial f_{\theta_k}^{(k)}}{\partial x_k}\right\|_2 \\ \Delta_k &\leq l_{k+1} \max_{x_k} \left\|\frac{\partial f_{\theta_k}^{(k)}}{\partial \theta_k}\right\|_2 \end{aligned} \tag{4}$$

By definition 3 and the triangle inequality, the sensitivity of the gradient $\frac{\partial \mathcal{L}_k}{\partial \theta_k}$ is upper bounded by twice $\max_{x_k} \|\frac{\partial \mathcal{L}_k}{\partial \theta_k}\|$, so $\Delta_k \geq s_2\left(\frac{\partial \mathcal{L}_k}{\partial \theta_k}\right)/2$.

Note that we can easily provide such upper bounds l_k and Δ_k if the layers $f_\theta^{(k)}$ and the loss ℓ are Lipschitz. If so, since all $f_\theta^{(k)}$ and ℓ are differentiable on any x_k, per Rademacher's theorem Rademacher (1919), $\left\|\frac{\partial \mathcal{L}_k}{\partial x_k}\right\|$ is bounded by the Lipschitz value of $\mathcal{L}_k$. We only need to find a tight upper bound of this Lipschitz value.

3.2 Estimating lipschitz values

In this section we bound Lipschitz values of different types of layers. We treat linear operations (e.g., linear transformations, convolutions) and activation functions as different layers.

Loss function and activation layer. Examples of Lipschitz losses encompass Softmax Cross-entropy, Cosine Similarity, and Multiclass Hinge. When it comes to activation layers, layers composed of an activation function, several prevalent ones, such as ReLU, tanh, and Sigmoid are 1-Lipschitz. We provide a detailed list in the Appendix Table 2.

Normalization layer. To be able to easily bound sensitivity, we define the operation of a normalization layer $f_{\theta_k}^{(k)}$ slightly differently than Eq (1):

$$x_{k+1}^{(k:q)} = f_{\theta_k}^{(k)}(x_k^{(k:q)}) = \frac{x_k^{(k:q)} - \mu^{(k:q)}}{\max(\alpha, \sigma^{(k:q)})}. \tag{5}$$

with α an hyperparameter. It is easy to see that the sensitivity is bounded by

$$\left\| \frac{\partial f_{\theta_k}^{(k)}}{\partial x_k} \right\|_2 \leq \max_{q \in [|\Gamma_k|]} \frac{1}{\max\left(\alpha, \sigma^{(k:q)}\right)} \leq 1/\alpha. \tag{6}$$

Note that a group normalization layer has no trainable parameters.

Linear layers. If $f_{\theta_k}^{(k)}$ is a linear layer, then

$$\begin{aligned} \left\| \frac{\partial f_{\theta_k}^{(k)}}{\partial \theta_k} \right\|_2 &= \left\| \frac{\partial (W_k^\top x_k + B_k)}{\partial (W_k, B_k)} \right\|_2 = \|(x_k, 1)\|_2, \\ \left\| \frac{\partial f_{\theta_k}^{(k)}}{\partial x_k} \right\|_2 &= \left\| \frac{\partial (W_k^\top x_k + B_k)}{\partial x_k} \right\|_2 = \|W_k\|_2. \end{aligned} \tag{7}$$

Convolutional layers. There are many types of convolutional layers, e.g., depending on the data type (strings, 2D images, 3D images ...), shape of the filter (rectangles, diamonds ...). Here we provide as an example only a derivation for convolutional layers for 2D images with rectangular filter. In that case, the input layer consists of $n_k = c_{in}hw$ nodes and the output layer consists of $n_{k+1} = c_{out}hw$ nodes with c_{in} input channels, c_{out} output channels, h the height of the image and w the width. Then, $\theta_k \in \mathbb{R}^{c_{in} \times c_{out} \times h' \times w'}$ with h' the height of the filter and w' the width of the filter. Indexing input and output with channel and coordinates, i.e., $x_k \in \mathbb{R}^{c_{in} \times h \times w}$ and $x_{k+1} \in \mathbb{R}^{c_{out} \times h \times w}$ we can then write

$$x_{k+1,c,i,j} = \sum_{d=1}^{c_{in}} \sum_{r=1}^{h'} \sum_{s=1}^{w'} x_{k,d,i+r,j+s} \theta_{k,c,d,r,s}$$

where components out of range are zero. We can derive (see Appendix B.1 for details) that

$$\left\| \frac{\partial f_{\theta_k}^{(k)}}{\partial x_k} \right\|_2 \leq \sqrt{h'w'} \|\theta_k\|_2 \tag{8}$$

$$\left\| \frac{\partial f_{\theta_k}^{(k)}}{\partial \theta_k} \right\|_2 \leq \sqrt{h'w'} \|x_k\|_2 \tag{9}$$

We summarize the upper bounds of the Lipschitz values, either on the input or on the parameters, for each layer type in the Appendix Table 2. We can conclude that networks for which the norms of the parameter vectors θ_k are bounded, are Lipschitz networks as introduced in Miyato et al. (2018), i.e., they are FNN for which each layer function $f^{(k)}_{\theta_k}$ is Lipschitz. We will denote by $\Theta_{\leq C}$ and by $\Theta_{=C}$ the sets of all paremeter vectors θ for f_θ such that $\|\theta_k\| \leq C$ and $\|\theta_k\| = C$ respectively, for $k = 1 \dots K$.

Computing sensitivty. We will denote by L_{θ_k} an upper bound for $\left\| \frac{\partial f^{(k)}_{\theta_k}}{\partial \theta_k} \right\|$ and by L_{x_k} an upper bound for $\left\| \frac{\partial f^{(k)}_{\theta_k}}{\partial x_k} \right\|$. We can now introduce Algorithm 1 to compute the sensitivity Δ_k of layer k. Here we denote by X_k the maximal possible norm of x_k, i.e., for all possible inputs x_k, $\|x_k\| = \left\| (f^{(k-1)}_{\theta_{k-1}} \circ \dots \circ f^{(1)}_{\theta_1})(x_1) \right\| \leq X_k$, with X_1 the norm on which we scale every input x_1. It capitalizes on a forward pass to compute the maximal input norms X_k — which depends on the previous layer range, in the case of normalization or activation layers, or on $u^\theta_{k-1} = \min(C, \|\tilde{\theta}_{k-1}\|)$ (see Algorithm 2), in the case of linear or convolutional layers (one can verify for each type of layer that u^θ_{k-1} is an upper bound for $\|x_k\|/\|x_{k-1}\|$) — and a backward pass applying Equation 4.

Algorithm 1 LayerSensitivity$(f, \theta, u^{(\theta)})$

Input: K layer feed-forward model f, parameters θ, parameter norm $u^{(\theta)}$, max input norm X_1, upper bound of loss Lipschitz value l_{K+1}.
for $k = 1$ **to** K **do** ▷ Forward pass
  if $\theta_k = \emptyset$ **then**
    if k-th layer is a normalization layer **then**
      $X_{k+1} \leftarrow \min(|x_{k+1}|^{\frac{1}{2}}, \frac{X_k}{\alpha})$
    else ▷ e.g., activation layer
      $X_{k+1} \leftarrow X_k L_{x_k}$
    end if
  else
    $X_{k+1} \leftarrow X_k u^{(\theta)}_k$
  end if
end for
for $k = K$ **to** 1 **do** ▷ Backward pass
  $l_k \leftarrow l_{k+1} L_{x_k}$
  $\Delta_k \leftarrow l_{k+1} L_{\theta_k}$
end for
Output: $(\Delta_1, \dots, \Delta_{K-1}, \Delta_K)$.

3.3 Lip-DP-SGD

We introduce a novel differentially private stochastic gradient descent algorithm, called Lip-DP-SGD, that leverages the estimation of the per-layer sensitivity of the model to provide differential privacy without gradient clipping.

Theorem 7. *Given a feed-forward model f_θ composed of Lipschitz constrained operators, a Lipschitz loss ℓ and a bounded input norm X_1,* Lip-DP-SGD *is differentially private.*

Indeed, Lip-DP-SGD utilizes the Gaussian mechanism. The gradient's sensitivity is determined without any privacy costs, as it depends only on the current parameter values (which are privatized in the previous step, and post-processing privatized values doesn't take additional privacy budget) and not on the data.

Algorithm 2 LIP-DP-SGD: Differentially Private Stochastic Gradient Descent with Lipschitz constrains.

Input: Data set $Z \in \mathcal{Z}^*$, feed-forward model f_θ, loss function $\mathcal{L}$, hypothesis space $\Theta \subseteq \mathbb{R}^k$, number of epochs T, noise multiplier σ, batch size $s \geq 1$, learning rate η, max gradient norm C
Initialize $\tilde{\theta}$ randomly from Θ
$(u^{(\theta)}, \tilde{\theta}) \leftarrow$ CLIPWEIGHTS($\tilde{\theta}$,C)
for $t \in [T]$ **do**
    $(\Delta_k)_{k=1}^K \leftarrow$ LAYERSENSITIVITY($f, \tilde{\theta}, u^{(\theta)}$)
    $V \leftarrow \emptyset$ ▷ Poisson sampling
    while $V = \emptyset$ **do**
        for $z \in Z$ **do**
            With probability $s/|Z|$: $V \leftarrow V \cup \{z\}$
        end for
    end while
    for $k = 1 \dots K$ **do** ▷ gradient per layer
        Draw $b_k \sim \mathcal{N}(0, \sigma^2 \Delta_k^2 \mathbb{I})$
        $\tilde{g}_k \leftarrow \frac{1}{|V|} \left(\sum_{i=1}^{|V|} \nabla_{\tilde{\theta}_k} \ell(f_{\tilde{\theta}}(x_i), y_i) + b_k \right)$
        $\tilde{\theta}_k \leftarrow \tilde{\theta}_k - \eta(t) \tilde{g}_k$
    end for
    $(u^{(\theta)}, \tilde{\theta}) \leftarrow$ CLIPWEIGHTS($\tilde{\theta}$,C)
end for
Output: $\tilde{\theta}$ and compute (ϵ, δ) with privacy accountant.
function CLIPWEIGHTS($\tilde{\theta}$, C)
    for $k = 1 \dots K$ **do**
        if $\theta_k \neq \emptyset$ **then**
            $u_k^{(\theta)} \leftarrow \min(C, \|\tilde{\theta}_k\|)$
            $\tilde{\theta}_k \leftarrow u_k^{(\theta)} \tilde{\theta}_k / \|\tilde{\theta}_k\|$
        end if
    end for
    return $(u^{(\theta)}, \tilde{\theta})$
end function

Privacy accounting. Lip-DP-SGD adopts the same privacy accounting as DP-SGD. Specifically, the accountant draws upon the privacy amplification Kasiviswanathan et al. (2011) brought about by Poisson sampling and the Gaussian moment accountant Abadi et al. (2016). It's worth noting that while we utilized the Renyi Differential Privacy (RDP) accountant Abadi et al. (2016); Mironov et al. (2019) in our experiments, Lip-DP-SGD is versatile enough to be compatible with alternative accountants.

Requirements. As detailed in Section 3.2, the loss and the model operators need to be Lipschitz and the norm of the input needs to be bounded. We've enumerated several losses and operators that meet these criteria in the Appendix Table 2. While we use the spectral norm to characterize Lipschitzness Yoshida and Miyato (2017); Miyato et al. (2018) in our study 3.2, other methods are also available, as discussed in Arjovsky et al. (2017).

ClipWeights. The CLIPWEIGHTS function is essential to the algorithm, ensuring Lipschitzness, which facilitates model sensitivity estimation. As opposed to standard Lipschitz-constrained networks Yoshida and Miyato (2017); Miyato et al. (2018) which increase or decrease the norms of parameters

to make them equal to a pre-definied value, our approach normalizes weights only when their current norm exceeds a threshold. This results in adding less DP noise for smaller norms. Importantly, as θ is already made private in the previous iteration, its norm is private too.

Computation techniques. For both Algorithm 1 and ClipWeights it's crucial to compute the greatest singular matrix values efficiently. A renowned technique is the *power method* von Mises and Pollaczek-Geiringer (1929). If this isn't sufficiently fast, the *power method* can be enhanced using AutoGrad Scaman and Virmaux (2019). Another idea is to use the Frobenius norm, which is faster to compute but may have drawbacks in terms of tightly bounding the norm. As computing spectral norms is relatively costly, we avoid to recompute them by storing them in $u^{(\theta)}$ in Algorithm 2.

3.4 Avoiding the bias of gradient clipping

Our Lip-DP-SGD algorithm finds a local optimum (for θ) of $F(\theta, Z)$ in $\Theta_{\leq C}$ while DP-SGD doesn't necessarily find a local optimum of $F(\theta, Z)$ in Θ. In particular, we prove in Appendix E the following

Theorem 8. *Let F be an objective function as defined in Section 2.2, and Z, f_θ, $\mathcal{L}$, $\Theta = \Theta_{\leq C}$, T, $\sigma = 0$, s, η and C be input parameters of Lip-DP-SGD satisfying the requirements specified in Section 3.3. Assume that for these inputs Lip-DP-SGD converges to a point θ^* (in the sense that $\lim_{k,T\to\infty} \theta_k = \theta^*$). Then, θ^* is a local optimum of $F(\theta, Z)$ in $\Theta_{\leq C}$.*

Essentially, making abstraction of the unbiased DP noise, the effect of scaling weight vectors to have bounded norm after a gradient step is equivalent to projecting the gradient on the boundary of the feasible space if the gradient brings the parameter vector out of $\Theta_{\leq C}$.

Furthermore, Chen et al. (2020) shows an example showing that gradient clipping can introduce bias. We add a more detailed discussion in Appendix E. Hence, DP-SGD does not necessarily converge to a local optimum of $F(\theta, Z)$, even when sufficient data is available to estimate θ. While Lip-DP-SGD can only find models in $\Theta_{\leq C}$ and this may introduce another suboptimality, as our experiments will show this is only a minor drawback in practice, while also others observed that Lipschitz networks have good properties Béthune et al. (2023). Moreover, it is easy to check whether Lip-DP-SGD outputs parameters on the boundary of $\Theta_{\leq C}$ and hence the model could potentially improve by relaxing the weight norm constraint. In contrast, it may not be feasible to detect that DP-SGD is outputting potentially suboptimal parameters. Indeed, consider a federated learning setting (e.g., Bonawitz et al. (2017)) where data owners collaborate to compute a model without revealing their data. Each data owner locally computes a gradient and clips it, and then the data owners securely aggregate their gradients and send the average gradient to a central party updating the model. In such setting, for privacy reasons no party would be able to evaluate that gradient clipping introduces a strong bias in some direction. Still, our experiments show that in practice at the time of convergence for the best hyperparameter values clipping is still active for a significant fraction of gradients (See Appendix C.5)

4 Experimental results

In this section, we conduct an empirical evaluation of our approach.

4.1 Experimental setup

We consider the following experimental questions:

Q1 How does Lip-DP-SGD, our proposed technique, compare against the conventional DP-SGD as introduced by Abadi et al. (2016)?

Q2 What is the effect of allowing $\|\theta_k\| < C$ rather than normalizing $\|\theta_k\|$ to C? This question seems relevant given that some authors (e.g., Béthune et al. (2023)) also suggest to consider networks which constant gradient norm rather than maximal gradient norm, i.e., roughly with θ in $\Theta_{=C}$ rather than $\Theta_{\leq C}$.

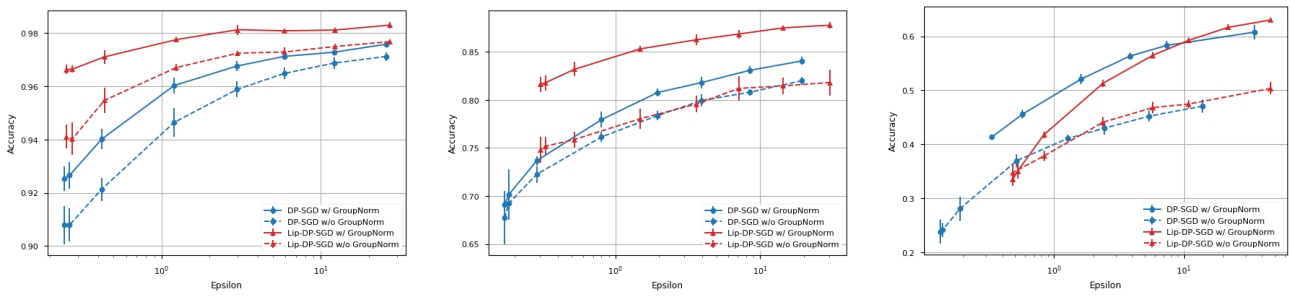
Implementation. We implemented both the DP-SGD and Lip-DP-SGD methods to ensure that comparisons were made under consistent model structures and preprocessing conditions.
To answer question Q2, we also implemented Fix-Lip-DP-SGD, a version of Lip-DP-SGD limited to networks whose weight norms are fixed, i.e., $\forall k : \|\theta_k\| = C$, obtained by setting $u_k^{(\theta)} \leftarrow C$ in Line 23 in Algorithm 2.

Hyperparameters. We selected a number of hyperparameters to tune for our experiments, aiming at making a fair comparison between the studied techniques while minimizing the distractions of potential orthogonal improvements. To optimize these hyperparameters, we used Bayesian optimization Balandat et al. (2020). Appendix C.1 provides a detailed discussion.

Datasets and models. We carried out experiments on both tabular datasets and datasets with image data. First, we consider a collection of 7 real-world tabular datasets (names and citations in Table 1). For these, we trained multi-layer perceptrons (MLP). A comprehensive list of model-dataset combinations is available in the Appendix Table 4.

Second, the image datasets include MNIST Deng (2012), Fashion-MNIST Xiao et al. (2017) and CIFAR-10 Krizhevsky et al. (2009). For these, we trained convolutional neural networks (CNN). We consider both networks with and without group normalization (see Section 3.2 or Wu and He (2018)). The networks using group normalization have normalization layers added after every convolutional layer. We opted for the accuracy to facilitate easy comparisons with prior research.

Infrastructure. All experiments were orchestrated across dual Tesla P100 GPU platforms (12GB capacity), operating under CUDA version 10, with a 62GB RAM provision for Fashion-MNIST and CIFAR-10. Remaining experiments were performed on an E5-2696V2 Processor setup, equipped with 8 vCPUs and a 52GB RAM cache. The total runtime of the experiments was approximately 50 hours, which corresponds to an estimated carbon emission of 1.96 kg Lacoste et al. (2019). More details on the experimental setup and an analysis of the runtime can be found in Appendix C.



(a) MNIST (b) Fashion-MNIST (c) CIFAR-10

Figure 1: Accuracy results, with a fixed $\delta = 10^{-5}$, for the MNIST (1(a)), Fashion-MNIST (1(b)), and CIFAR-10 (1(c)) datasets, comparing Lip-DP-SGD (in red) and DP-SGD (in blue), lines are dashed when the underlying model is implemented without group normalization. Vertical lines represent the standard error of the mean. See Appendix C.1 for details on model specifications and hyperparameters.

Table 1: Accuracy and ϵ per dataset and method at $\delta = 1/n$, in bold the best result and underlined when the difference with the best result is not statistically significant at a level of confidence of 5%.

Methods Datasets (#instances n × #features p)	ϵ	DP-SGD	Lip-DP-SGD	Fix-Lip-DP-SGD
Adult Income (48842x14) Becker et al. (1996)	0.414	0.824	**0.831**	0.804
Android (29333x87) Mathur et al. (2022)	1.273	0.951	**0.959**	0.945
Breast Cancer (569x32) Wolberg et al. (1995)	1.672	0.773	**0.798**	0.7
Default Credit (30000x24) Yeh (2016)	1.442	0.809	**0.816**	0.792
Dropout (4424x36) Realinho et al. (2021)	1.326	0.763	**0.819**	0.736
German Credit (1000x20) Hofmann (1994)	3.852	0.735	<u>0.746</u>	0.768
Nursery (12960x8) Rajkovic (1997)	1.432	0.919	**0.931**	0.89

4.2 Results

Image datasets. In Figure 1, Lip-DP-SGD demonstrates comparable or superior performance to DP-SGD for MNIST, Fashion-MNIST, and CIFAR-10 when not combined with group normalization. The inclusion of the normalization technique enhances the accuracy of DP-SGD, as discovered by De et al. (2022), and Lip-DP-SGD, with the latter experiencing a more significant improvement, except for CIFAR-10. It is noteworthy that other explored regularization techniques by De et al. (2022) enhance our method but to a similar extent as DP-SGD, see Appendix C.6

Tabular datasets. In Table 1, we perform a Wilcoxon Signed-rank test, at a confidence level of 5%, on 10 measures of accuracy for each dataset between the DP-SGD based on the gradient clipping and the Lip-DP-SGD based on our method. Lip-DP-SGD consistently outperforms DP-SGD in terms of accuracy. This trend holds across datasets with varying numbers of instances and features, including tasks with imbalanced datasets like Dropout or Default Credit datasets. While highly impactful for convolutional layers, group normalization does not yield improvements for either DP-SGD or Lip-DP-SGD in the case of tabular datasets. See Appendix C.4 for complete results.

Additionally, Table 1 presents the performance achieved by constraining networks to Lipschitz networks, where the norm of weights is set to a constant, denoted as Fix-Lip-DP-SGD. The results from this approach are inferior, even when compared to DP-SGD.

Conclusion. In summary, we can conclude that we can answer to our experimental questions that Lip-DP-SGD outperforms DP-SGD on both tabular data sets with MLP and image data sets with CNN. Moreover, it is beneficial to not normalize the norm of the weight vector θ to a fixed value but to exploit cases where it becomes smaller.

5 Related Work

DP-SGD. DP-SGD algorithms have been developped to guarantee privacy on the final output Chaudhuri et al. (2011), on the loss function Kifer et al. (2012) or on the publishing of each gradient used in the descent Bassily et al. (2014); Abadi et al. (2016).

To keep track of the privacy budget consumption, Bassily et al. (2014) relies on the strong composition theorem Dwork et al. (2010) while Abadi et al. (2016) is based on the moment accountant and gives much tighter bounds on the privacy loss than Bassily et al. (2014).

This has opened an active field of research that builds upon Abadi et al. (2016) in order to provide better estimation of the hyperparameters e.g., the clipping norm McMahan et al. (2017); Andrew et al. (2022), the learning rate Koskela and Honkela (2020), or the step size of the privacy budget

consumption Lee and Kifer (2018); Chen and Lee (2020); Yu et al. (2019); or to enhance performance with regularization techniques De et al. (2022). Gradient clipping remains the standard approach, and most of these ideas can be combined with our improvements, the most beneficial combination being the use of group normalization (Section 4.2).

Lipschitz continuity. Lipschitz continuity is an essential requirement for differential privacy in some private SGD algorithms Bassily et al. (2014). However, since deep neural networks (DNNs) have an unbounded Lipschitz value Scaman and Virmaux (2019), it is not possible to use it to scale the added noise. Several techniques have been proposed to enforce Lipschitz continuity to DNNs, especially in the context of generative adversarial networks (GANs) Miyato et al. (2018); Gouk et al. (2020). These techniques, which mainly rely on weight spectral normalization, can be applied to build DP-SGD instead of the gradient clipping method, as described in Section 3. Bethune et al. (2023) suggests a number of ideas in the direction of our Fix-Lip-DP-SGD variant, but has only limited empirical evaluation. Our paper shows that Lip-DP-SGD outperforms Fix-Lip-DP-SGD (Table 1), and highlights the great synergy between weight normalization and group normalization (Figure 1).

6 Conclusion and discussion

In this paper we proposed a new differentially private stochastic gradient descent algorithm without gradient clipping. We derived a methodology to estimate the gradient sensitivity to scale the noise. An important advantage of weight clipping over gradient clipping is that it avoids the bias introduced by gradient clipping and the algorithm converges to a local optimum of the objective function. We showed empirically that this yields a significant improvement in practice and we argued that this approach circumvent the bias induced by classical gradient clipping.

Several opportunities for future work remain. First, it would be interesting to better integrate and improve ideas such as in Scaman and Virmaux (2019) to find improved bounds on gradients of Lipschitz-constrained neural networks, as this may allow to further reduce the amount of noise needed.

Second, various optimizations of the computational efficiency are possible. Currently one of the most important computational tasks is the computation of the spectral norm. Other approaches to more efficiently compute or upper bound it can be explored. One alternative direction would be to investigate the Frobenius norm which is less costly to compute but may have other disadvantages.

Our current work is limited to the application of our proposed method on feed-forward models for classification tasks and regression tasks with Lipschitz loss function. Although our method can be easily applied to some other tasks, the field remains open to extend it to other classes of models.

Finally, while our experiments have shown promising results, further theoretical analysis of Lip-DP-SGD, especially the interaction between sensitivity, learning rate and number of iterations, remains an interesting area of research, similar to the work of Song et al. (2020) on DP-SGD. An analysis on the interactions between hyperparameters would provide valuable insights into the optimal use of our method and its potential combination with other regularization techniques.

Acknowledgements

This work was partially supported by the Horizon Europe TRUMPET project grant no 101070038 and the ANR PMR project grant no ANR-20-CE23-0013.

References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. (2016). Deep Learning with Differential Privacy. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318. arXiv: 1607.00133.

Andrew, G., Thakkar, O., McMahan, H. B., and Ramaswamy, S. (2022). Differentially Private Learning with Adaptive Clipping. In *Advances in Neural Information Processing Systems*. arXiv. arXiv:1905.03871 [cs, stat].

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 214–223. JMLR.org.

Balandat, M., Karrer, B., Jiang, D. R., Daulton, S., Letham, B., Wilson, A. G., and Bakshy, E. (2020). BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. arXiv:1910.06403 [cs, math, stat].

Bassily, R., Feldman, V., Talwar, K., and Guha Thakurta, A. (2019). Private Stochastic Convex Optimization with Optimal Rates. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Bassily, R., Smith, A., and Thakurta, A. (2014). Private empirical risk minimization: Efficient algorithms and tight error bounds. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 464–473.

Becker, Barry, Kohavi, and Ronny (1996). Adult. UCI Machine Learning Repository. DOI: `10.24432/C5XW20`.

Bethune, L., Massena, T., Boissin, T., Prudent, Y., Friedrich, C., Mamalet, F., Bellet, A., Serrurier, M., and Vigouroux, D. (2023). Dp-sgd without clipping: The lipschitz neural network way.

Béthune, L., Novello, P., Coiffier, G., Boissin, T., Serrurier, M., Vincenot, Q., and Troya-Galvis, A. (2023). Robust one-class classification with signed distance function using 1-lipschitz neural networks. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. (2017). Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 1175–1191, New York, NY, USA. Association for Computing Machinery.

Chaudhuri, K., Monteleoni, C., and Sarwate, A. D. (2011). Differentially Private Empirical Risk Minimization. *Journal of Machine Learning Research*, page 41.

Chen, C. and Lee, J. (2020). Stochastic adaptive line search for differentially private optimization. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 1011–1020.

Chen, X., Wu, S. Z., and Hong, M. (2020). Understanding gradient clipping in private sgd: A geometric perspective. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 13773–13782. Curran Associates, Inc.

Daulton, S., Balandat, M., and Bakshy, E. (2020). Differentiable Expected Hypervolume Improvement for Parallel Multi-Objective Bayesian Optimization. arXiv:2006.05078 [cs, math, stat].

De, S., Berrada, L., Hayes, J., Smith, S. L., and Balle, B. (2022). Unlocking High-Accuracy Differentially Private Image Classification through Scale. *arXiv preprint arXiv:2204.13650*.

Deng, L. (2012). The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. (2006). Calibrating Noise to Sensitivity in Private Data Analysis. In Halevi, S. and Rabin, T., editors, *Theory of Cryptography*, pages 265–284, Berlin, Heidelberg. Springer Berlin Heidelberg.

Dwork, C. and Roth, A. (2013). The Algorithmic Foundations of Differential Privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3-4):211–407.

Dwork, C., Rothblum, G. N., and Vadhan, S. (2010). Boosting and Differential Privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 51–60, Las Vegas, NV, USA. IEEE.

Gouk, H., Frank, E., Pfahringer, B., and Cree, M. J. (2020). Regularisation of neural networks by enforcing lipschitz continuity.

Hofmann, H. (1994). Statlog (German Credit Data). UCI Machine Learning Repository. DOI: 10.24432/C5NC77.

Ioffe, S. and Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *ICML*.

Kasiviswanathan, S. P., Lee, H. K., Nissim, K., Raskhodnikova, S., and Smith, A. (2011). What can we learn privately? *SIAM Journal on Computing*, 40(3):793–826.

Kifer, D., Smith, A., and Thakurta, A. (2012). Private Convex Empirical Risk Minimization and High-dimensional Regression. In *Proceedings of the 25th Annual Conference on Learning Theory*, pages 25.1–25.40. JMLR Workshop and Conference Proceedings. ISSN: 1938-7228.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Koskela, A. and Honkela, A. (2020). Learning Rate Adaptation for Differentially Private Learning. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, pages 2465–2475. PMLR. ISSN: 2640-3498.

Krizhevsky, A., Nair, V., and Hinton, G. (2009). Cifar-10 (canadian institute for advanced research). *URL http://www. cs. toronto. edu/kriz/cifar. html*.

Lacoste, A., Luccioni, A., Schmidt, V., and Dandres, T. (2019). Quantifying the carbon emissions of machine learning.

Lee, J. and Kifer, D. (2018). Concentrated differentially private gradient descent with adaptive per-iteration privacy budget. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, page 1656–1665, New York, NY, USA. Association for Computing Machinery.

Mathur, Akshay, Mathur, and Akshay (2022). NATICUSdroid (Android Permissions) Dataset. UCI Machine Learning Repository.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and Arcas, B. A. y. (2017). Communication-Efficient Learning of Deep Networks from Decentralized Data. In Singh, A. and Zhu, J., editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR.

Mironov, I., Talwar, K., and Zhang, L. (2019). Rényi Differential Privacy of the Sampled Gaussian Mechanism. *arXiv e-prints*, page arXiv:1908.10530.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. (2018). Spectral normalization for generative adversarial networks. In *Advances in Neural Information Processing Systems*.

Papernot, N. and Steinke, T. (2022). Hyperparameter Tuning with Renyi Differential Privacy. In *International Conference on Learning Representations*. arXiv. arXiv:2110.03620 [cs].

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Rademacher, H. (1919). Über partielle und totale differenzierbarkeit von funktionen mehrerer variabeln und über die transformation der doppelintegrale. *Mathematische Annalen*, 79:340–359.

Rajkovic, V. (1997). Nursery. UCI Machine Learning Repository. DOI: 10.24432/C5P88W.

Realinho, V., Vieira Martins, M., Machado, J., and Baptista, L. (2021). Predict students' dropout and academic success. UCI Machine Learning Repository. DOI: 1.

Scaman, K. and Virmaux, A. (2019). Lipschitz regularity of deep neural networks: analysis and efficient estimation. In *Advances in Neural Information Processing Systems*. arXiv. arXiv:1805.10965 [cs, stat].

Song, S., Steinke, T., Thakkar, O., and Thakurta, A. G. (2020). Evading the curse of dimensionality in unconstrained private generalized linear problems. In *Proceedings of the 23th International Conference on Artificial Intelligence and Statistics*.

von Mises, R. and Pollaczek-Geiringer, H. (1929). Praktische verfahren der gleichungsauflösung . *Zamm-zeitschrift Fur Angewandte Mathematik Und Mechanik*, 9:58–77.

Wolberg, William, Mangasarian, Olvi, Street, Nick, and Street, W. (1995). Breast Cancer Wisconsin (Diagnostic). UCI Machine Learning Repository. DOI: 10.24432/C5DW2B.

Wu, Y. and He, K. (2018). Group normalization. *International Journal of Computer Vision*, 128:742 – 755.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. arXiv:1708.07747 [cs, stat].

Yeh, I.-C. (2016). default of credit card clients. UCI Machine Learning Repository. DOI: 10.24432/C55S3H.

Yoshida, Y. and Miyato, T. (2017). Spectral Norm Regularization for Improving the Generalizability of Deep Learning. arXiv:1705.10941 [cs, stat].

Yu, L., Liu, L., Pu, C., Gursoy, M. E., and Truex, S. (2019). Differentially Private Model Publishing for Deep Learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 332–349. arXiv:1904.02200 [cs].

Ziller, A., Usynin, D., Knolle, M., Prakash, K., Trask, A., Braren, R., Makowski, M., Rueckert, D., and Kaissis, G. (2021). Sensitivity analysis in differentially private machine learning using hybrid automatic differentiation. arXiv:2107.04265 [cs].

A Gradient clipping based DP-SGD

For comparison with Algorithm 2, Algorithm 3 shows the classic DP-SGD algorithm based on gradient clipping.

Algorithm 3 DP-SGD: Differentially Private Stochastic Gradient Descent with gradient clipping.

Input: Data set $Z \in \mathcal{Z}^*$, model f_θ, loss function $\mathcal{L}$, hypothesis space $\Theta \subseteq \mathbb{R}^k$, privacy parameters ϵ and δ, noise multiplier σ, batch size $s \geq 1$, learning rate η, max gradient norm C
Initialize $\tilde{\theta}$ randomly from Θ
for $t \in [T]$ **do**
  $V \leftarrow \emptyset$ ▷ Poisson sampling
  while $S = \emptyset$ **do**
    for $z \in Z$ **do**
      With probability $s/|Z|$: $V \leftarrow V \cup \{z\}$
    end for
  end while
  Draw $b_k \sim \mathcal{N}(0, \sigma^2 C^2 \mathbb{I})$
  for $i = 1 \dots |V|$ **do** ▷ Gradient clipping per sample
    $\tilde{g}_{k,i} \leftarrow \nabla_{\tilde{\theta}_k} \ell(f_{\tilde{\theta}}(x_i)) \min(1, C/\|\nabla_{\tilde{\theta}_k} \mathcal{L}(f(x_i; \tilde{\theta}))\|)$
  end for
  $\tilde{g}_k \leftarrow \frac{1}{|V|} \left(\sum_{i=1}^{|V|} \tilde{g}_{k,i} + b_k \right)$
  $\tilde{\theta}_k \leftarrow \tilde{\theta}_k - \eta(t) \tilde{g}_k$
end for
Output: $\tilde{\theta}$ and compute privacy cost (ϵ, δ) with privacy accountant.

B Estimating Lipschitz values

We summarize the upper bounds of the Lipschitz values, either on the input or on the parameters, for each layer type in Table 2. It's important to mention that for the loss, the Lipschitz value is solely dependent on the output x_{K+1}.

Layer	Definition	Lip. value on input x_k	Lip. value on parameter θ_k
Dense	$\theta_k^\top x_k$	$\|\theta_k\|$	$\|x_k\|$
Convolutional	$\theta_k * x_k$	$\sqrt{h'w'}\|\theta_k\|$	$\sqrt{h'w'}\|x_k\|$
Normalization	$\frac{x_k^{(k:q)} - \mu^{(k:q)}}{\max(\alpha, \sigma^{(k:q)})}$	$1/\alpha$	-
ReLU	$\max(x_k, 0)$	1	-
Sigmoid	$\frac{1}{1+e^{-x_k}}$	$1/2$	-
Softmax Cross-entropy	$y \log \textsc{softmax}(x_{K+1})/\tau$	$\sqrt{2}/\tau$	-
Cosine Similarity	$\frac{x_{K+1}^\top y}{\|x_{K+1}\|_2 \|y\|_2}$	$1/\min \|x_{K+1}\|$	-
Multiclass Hinge	$\left\{\max\left(0, \frac{m}{2} - x_{i_{K+1}} \cdot y_i\right)\right\}$	1	-

Table 2: Summary table of Lipschitz values with respect to the layer.

with $\textsc{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{c} \exp(x_j)}$, c the number of classes. For cross-entropy, τ an hyperparameter on the Softmax Cross-entropy loss also known as the temperature. For convolutional layers, h' and w' are the height and width of the filter. For multiclass hinge, m is a hyperparameter known as 'margin'.

B.1 Details for the convolutional layer

Theorem 9. *The convolved feature map* $(\theta * \cdot) : \mathbb{R}^{n_k \times |x_k|} \to \mathbb{R}^{n_{k+1} \times n \times n}$*, with zero or circular padding, is Lipschitz and*

$$\|\nabla_{\theta_k}(\theta_k * x_k)\|_2 \leq \sqrt{h'w'}\|x_k\|_2 \text{ and } \|\nabla_{x_k}(\theta_k * x_k)\|_2 \leq \sqrt{h'w'}\|\theta_k\|_2 \tag{10}$$

with w' *and* h' *the width and the height of the filter.*

Proof. The output $x_{k+1} \in \mathbb{R}^{c_{out} \times n \times n}$ of the convolution operation is given by:

$$x_{k+1,c,r,s} = \sum_{d=0}^{c_{in}-1} \sum_{i=0}^{h'-1} \sum_{j=0}^{w'-1} x_{k,d,r+i,s+j}\theta_{k,c,d,i,j}$$

There follows:

$$\begin{aligned}
\|x_{k+1}\|_2^2 &= \sum_{c=0}^{c_{out}-1} \sum_{r=1}^{n} \sum_{s=1}^{n} \left(\sum_{d=0}^{c_{in}-1} \sum_{i=0}^{h'-1} \sum_{j=0}^{w'-1} x_{k,d,r+i,s+j}\theta_{k,c,d,i,j} \right)^2 \\
&\leq \sum_{c=0}^{c_{out}-1} \sum_{r=1}^{n} \sum_{s=1}^{n} \left(\sum_{d=0}^{c_{in}-1} \sum_{i=0}^{h'-1} \sum_{j=0}^{w'-1} x_{k,d,r+i,s+j}^2 \right) \left(\sum_{d=0}^{c_{in}-1} \sum_{i=0}^{h'-1} \sum_{j=0}^{w'-1} \theta_{k,c,d,i,j}^2 \right) \\
&= \left(\sum_{d=0}^{c_{in}-1} \sum_{i=0}^{h'-1} \sum_{j=0}^{w'-1} \sum_{r=1}^{n} \sum_{s=1}^{n} x_{k,d,r+i,s+j}^2 \right) \left(\sum_{c=0}^{c_{out}-1} \sum_{d=0}^{c_{in}-1} \sum_{i=0}^{h'-1} \sum_{j=0}^{w'-1} \theta_{k,c,d,i,j}^2 \right) \\
&\leq h'w' \left(\sum_{d=0}^{c_{in}-1} \sum_{r=1}^{n} \sum_{s=1}^{n} x_{k,d,r,s}^2 \right) \left(\sum_{c=0}^{c_{out}-1} \sum_{d=0}^{c_{in}-1} \sum_{i=0}^{h'-1} \sum_{j=0}^{w'-1} \theta_{k,c,d,i,j}^2 \right) \\
&= h'w'\|x_k\|_2\|\theta_k\|_2
\end{aligned}$$

Since $\theta_k * \cdot$ is a linear operator:

$$\|(\theta_k * x_k) - (\theta'_k * x_k)\|_2 = \|(\theta_k - \theta'_k) * x_k\|_2 \leq \|\theta_k - \theta'_k\|_2\sqrt{h'w'}\|x_k\|_2$$

Finally, the convolved feature map is differentiable so the spectral norm of its Jacobian is bounded by its Lipschitz value:

$$\|\nabla_{\theta_k}(\theta_k * x_k)\|_2 \leq \sqrt{h'w'}\|x_k\|_2$$

Analogously,

$$\|\nabla_{x_k}(\theta_k * x_k)\|_2 \leq \sqrt{h'w'}\|\theta_k\|_2$$

□

C Experiments

Optimization. For both tabular and image datasets, we employed Bayesian optimization Balandat et al. (2020). Configured as a multi-objective optimization program Daulton et al. (2020), our focus was to cover the Pareto front between model utility (accuracy) and privacy (ϵ values at a constant level of δ, set to $1/n$ as has become common in this type of experiments). To get to finally reported values, we select the point on the pareto front given by the Python librairy BoTorch Balandat et al. (2020).

C.1 Hyperparameters

Hyperparameter selection. In the literature, there are a wide range of improvements possible over a direct application of SGD to supervised learning, including general strategies such as pre-training, data augmentation and feature engineering, and DP-SGD specific optimizations such as adaptive maximum gradient norm thresholds. All of these can be applied in a similar way to both Lip-DP-SGD and DP-SGD and to keep our comparison sufficiently simple, fair and understandable we didn't consider the optimization of these choices.

We did tune hyperparameters inherent to specific model categories, in particular the initial learning rate $\eta(0)$ (to start the adaptive learning rate strategy $\eta(t)$) and (for image datasets) the number of groups, and hyperparameters related to the learning algorithm, in particular the (expected) batch size s, the Lipschitz upper bound of the normalization layer α and the threshold C on the gradient norm respectively weight norm.

The initial learning rate $\eta(0)$ is tuned while the following $\eta(t)$ are set adaptively. Specifically, we use the strategy of the Adam algorithm Kingma and Ba (2014), which update each parameter using the ratio between the moving average of the gradient (first moment) and the square root of the moving average of its squared value (second moment), ensuring fast convergence.

We also investigated varying the maximum norm of input vectors X_0 and the hyperparameter τ of the cross entropy objective function, but the effect of these hyperparameters turned out to be insignificant.

Both the clipping threshold C for gradients in DP-SGD and the clipping threshold C for weights in Lip-DP-SGD can be tuned for each layer separately. While this offers improved performance, it does come with the cost of consuming more of the privacy budget, and substantially increasing the dimensionality of the hyperparameter search space. In a few experiments we didn't see significant improvements in allowing per-layer varying of C_k, so we didn't further pursue this avenue.

Table 3 summarizes the search space of hyperparameters. It's important to note that we did not account for potential (small) privacy losses caused by hyperparameter search, a limitation also acknowledged in other recent works such as Papernot and Steinke (2022).

Hyperparameter	Range
Noise multiplier σ	[0.4, 5]
Weight clipping threshold C	[1, 15]
Gradient clipping threshold C	[1, 15]
Batch size s	[32, 512]
$\eta(0)$	[0.0001, 0.01]
Number of groups (group normalization)	[8, 32]
α (group normalization)	$[0.1/(\lvert x_k^{(k:q)}\rvert), 1/(\lvert x_k^{(k:q)}\rvert)]$

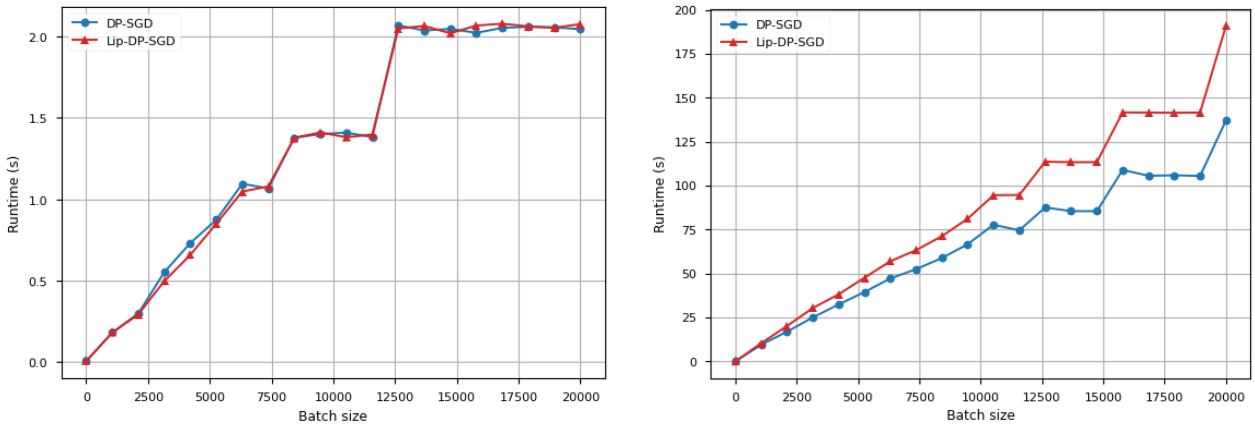
Table 3: Summary of hyperparameter space.

C.2 Models

Table 4 shows details of the models we used to train on tabular and image datasets. We consider 7 tabular datasets: adult income Becker et al. (1996), android permissions Mathur et al. (2022), breast cancer Wolberg et al. (1995), default credit Yeh (2016), dropout Realinho et al. (2021), German credit Hofmann (1994) and nursery Rajkovic (1997). See Table 1 for the number of instances and features for each tabular dataset.

Dataset	Image size	Model	Number of layers	Loss	No. of Parameters
Tabular Datasets	-	MLP	2	CE	140 to 2,120
MNIST	28x28x1	ConvNet	3	CE	83,154
FashionMNIST	28x28x1	ConvNet	6	CE	132,746
CIFAR-10	32x32x3	ShallowVGG	6	CE	131,466

Table 4: Summary table of datasets with respective models architectures details.



(a) Default Credit

(b) MNIST

Figure 2: Mean runtime in seconds per batch size on one epoch over the Default Credit dataset 2(a) and the MNIST dataset 2(b) comparing DP-SGD (in blue) and Lip-DP-SGD (in red).

C.3 Runtime

Our experiments didn't show significant deviations from the normal runtime behavior one can expect for neural network training. As an illustration, we compared on the Default Credit dataset and on the MNIST dataset the mean epoch runtime of DP-SGD with Lip-DP-SGD. We measure runtime against the logical batch size, limiting the physical batch size to prevent memory errors as recommended by the PyTorch documentation Paszke et al. (2019). Figure 2 shows how Lip-DP-SGD is slightly inefficient in terms of runtime compared to DP-SGD, especially on image datasets. It may be possible to further improve Lip-DP-SGD runtime as it currently heavily relies on the data sampler provided by Opacus, which processes data per instance, while applying batch processing techniques inspired on PyTorch would be more efficient. The staircase shape of the plot seems to be a result of PyTorch and Python memory management strategies.

C.4 Detailed results

Table 5 provides a summary of accuracy performances for tabular datasets with and without group normalization. It's worth noting that while epsilon values may not be identical across algorithms, we present the epsilon value of DP-SGD and report performances corresponding to lower epsilon values for the other two algorithms, consistent with Table 1.

Table 5: Accuracy per dataset and method at $\epsilon = 1$ and $\delta = 1/n$, in bold the best result and underlined when the difference with the best result is not statistically significant at a level of confidence of 5%.

Methods		DP-SGD		Lip-DP-SGD		Fix-Lip-DP-SG	
Datasets (#instances n × #features p)	ϵ	w/ GN	w/o GN	w/ GN	w/o GN	w/ GN	w/o G
Adult Income (48842x14) Becker et al. (1996)	0.414	0.822	0.824	0.829	**0.831**	0.713	0.8
Android (29333x87) Mathur et al. (2022)	1.273	0.947	0.951	0.952	**0.959**	0.701	0.9
Breast Cancer (569x32) Wolberg et al. (1995)	1.672	0.813	0.773	**0.924**	0.798	0.519	
Default Credit (30000x24) Yeh (2016)	1.442	0.804	0.809	0.815	**0.816**	0.774	0.7
Dropout (4424x36) Realinho et al. (2021)	1.326	0.755	0.763	0.816	**0.819**	0.573	0.7
German Credit (1000x20) Hofmann (1994)	3.852	0.735	0.735	0.722	<u>0.746</u>	0.493	0
Nursery (12960x8) Rajkovic (1997)	1.432	0.916	0.919	0.912	**0.931**	0.487	0

C.5 Gradient clipping behavior

In Section 3.4 we argued that DP-SGD introduces bias. There are several ways to demonstrate this. For illustration we show here the error between the true average gradient

$$g_k^{Lip-DP-SGD} = \frac{1}{|V|} \sum_{i=1}^{|V|} \nabla_{\tilde{\theta}_k} \ell(f_{\tilde{\theta}}(x_i))$$

i.e., the model update of Algorithm 2 without noise, and the average clipped gradient

$$g_k^{DP-SGD} = \frac{1}{|V|} \sum_{i=1}^{|V|} \text{clip}_C \left(\nabla_{\tilde{\theta}_k} \ell(f_{\tilde{\theta}}(x_i)) \right),$$

i.e., the model update of Algorithm 3 without noise.

Figure 3 shows the error $\left\| g_k^{Lip-DP-SGD} - g_k^{DP-SGD} \right\|$ together with the norm of the DP-SGD model update $\left\| g_k^{DP-SGD} \right\|$.

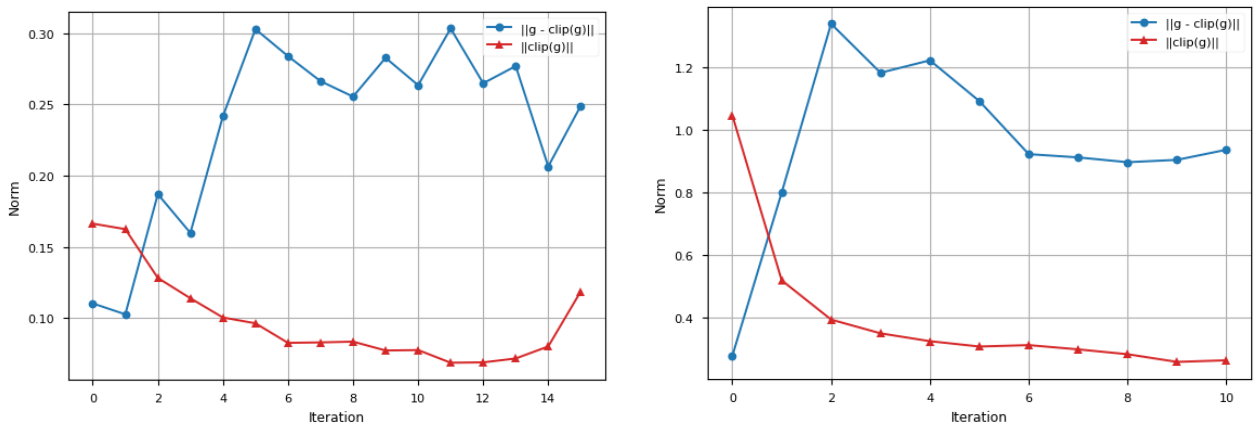
One can observe for both considered datasets that while the model converges and the average clipped gradient decreases, the error between DP-SGD's average clipped gradient and the true average gradient increases. At the end, the error in the gradient caused by clipping is significant, and hence the model converges to a different point than the real optimum.

C.6 Regularization techniques.

In De et al. (2022) multiple other regularization/optimization techniques are proposed. In our experiments, we found that while these techniques sometimes help, there is no significant or systematic difference in the effect on the results of DP-SGD and Lip-DP-SGD. For example, Figure 4 illustrates how parameter averaging on Fashion-MNIST, as indicated by De et al. (2022), has a similar impact observed for both DP-SGD and Lip-DP-SGD. To keep our experiments simple and interpretable, in all other experiments in this paper we don't consider the optimizations proposed by De et al. (2022), except for the group normalization.

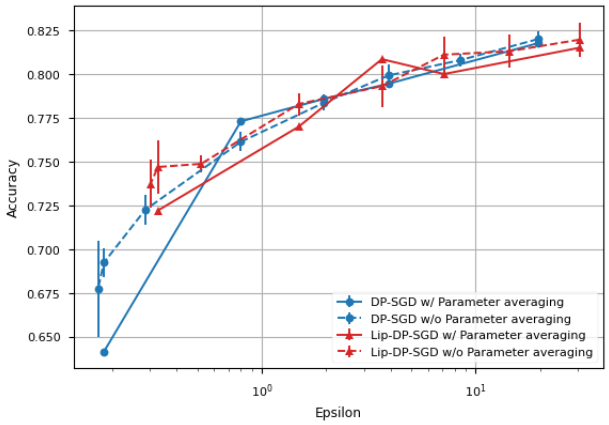
D Lip-DP-SGD library

We offer an open-source toolkit for implementing LipDP-SGD on any FNN model structure. This toolkit builds on the Opacus and PyTorch libraries. Drawing inspiration from Opacus, our library is based on two main components: the 'DataLoader', which utilizes Poisson sampling to harness the



(a) Dropout (b) Adult Income

Figure 3: Norm of the average error $g - \text{clip}(g)$ (in blue) and norm of the average of $\text{clip}(g)$ (in red) across training iterations on the Dropout dataset 3(a) (averaged over 500 instances) and the Adult Income dataset 3(b) (averaged over 500 instances).



(a) Fashion-MNIST

Figure 4: Accuracy results for the Fashion-MNIST, comparing DP-SGD (in blue) and Lip-DP-SGD (in red), lines are dashed when the underlying model is implemented without parameter averaging.

advantages of privacy amplification Kasiviswanathan et al. (2011), and the 'Optimizer', responsible for sensitivity calculation, differential privacy noise addition, and parameter normalization during each iteration.

'README.md', provided in the supplementary materials, details how to run the library and how to reproduce the experiments.

E Avoiding the bias of gradient clipping

We show that Lip-DP-SGD converges to a local minimum in $\Theta_{\leq C}$ while DP-SGD suffers from bias and may converge to a point which is not a local minimum of Θ.

We use the word 'converge' here somewhat informally, as in each iteration independent noise is added the objective function slightly varies between iterations and hence none of the mentioned algorithms converges to an exact point. We here informally mean approximate convergence to a small region, assuming a sufficiently large data set Z and/or larger ϵ such that privacy noise doesn't significantly alter the shape of the objective function. Our argument below hence makes abstraction of the noise for simplicity, but in the presence of small amounts of noise a similar argument holds approximately, i.e., after sufficient iterations Lip-DP-SGD will produce θ values close to a locally optimal θ^* while DP-SGD may produce θ values in a region not containing the relevant local minimum.

First, let us consider convergence.

Theorem 8. Let F be an objective function as defined in Section 2.2, and Z, f_θ, $\mathcal{L}$, $\Theta = \Theta_{\leq C}$, T, $\sigma = 0$, s, η and C be input parameters of Lip-DP-SGD satisfying the requirements specified in Section 3.3. Assume that for these inputs Lip-DP-SGD converges to a point θ^* (in the sense that $\lim_{k,T\to\infty} \theta_k = \theta^*$). Then, θ^* is a local optimum of $F(\theta, Z)$ in $\Theta_{\leq C}$.

Proof sketch. We consider the problem of finding a local optimum in $\Theta_{\leq C}$:

$$\begin{array}{ll} \text{minimize} & F(\theta, Z) \\ \text{subject to} & \|\theta\|_2 \leq C \end{array}$$

We introduce a slack variable ζ:

$$\begin{array}{ll} \text{minimize} & F(\theta, Z) \\ \text{subject to} & \|\theta\|_2 + \zeta^2 = C \end{array}$$

Using Lagrange multipliers, we should minimize

$$F(\theta, Z) - \lambda(\|\theta\|_2 + \zeta^2 - C)$$

An optimum in θ, λ and ζ satisfies

$$\nabla_\theta F(\theta, Z) - \lambda\theta = 0 \quad (11)$$

$$\|\theta\|_2 + \zeta^2 - C = 0 \quad (12)$$

$$2\lambda\zeta = 0 \quad (13)$$

From Eq 13, either $\lambda = 0$ or $\zeta = 0$ If $\zeta > 0$, θ is in the interior of $\Theta_{\leq C}$ and there follows $\lambda = 0$ and from Eq 11 that $\nabla_\theta F(\theta, Z) = 0$. For such θ, Lip-DP-SGD does not perform weight clipping. If the learning rate is sufficiently small, and if it converges to a θ with norm $\|\theta\|_2 < C$ it is a local optimum. On the other hand, if $\zeta = 0$, there follows from Eq 12 that $\|\theta\|_2 = C$, i.e., θ is on the boundary of $\Theta_{\leq C}$. If θ is a local optimum in $\Theta_{\leq C}$, then $\nabla_\theta F(\theta, Z)$ is perpendicular on the ball of vectors θ with norm C, and for such θ Lip-DP-SGD will add the multiple $\eta(t).\nabla_\theta F(\theta, Z)$ to θ and will next scale θ back to norm C, leaving θ unchanged. For a θ which is not a local optimum in $\Theta_{\leq C}$, $\nabla_\theta F(\theta, Z)$ will not be perpendicular to the ball of C-norm parameter vectors, and adding the gradient and brining the

norm back to C will move θ closer to a local optimum on this boundary of $\Theta_{\leq C}$. This is consistent with Eq 11 which shows the gradient with respect to θ for the constrained problem to be of the form $\nabla_\theta F(\theta, Z) - \lambda\theta$. □

Theorem 8 shows that in a noiseless setting, if Lip-DP-SGD converges to a stable point that point will be a local optimum in $\Theta_{\leq C}$. In the presence of noise and/or stochastic batch selection, algorithms of course don't converge to a specific point but move around close to the optimal point due to the noise in each iteration, and advanced methods exist to examine such kind of convergence. The conclusion remains the same: Lip-DP-SGDwill converge to a neighborhood of the real local optimum, while as we argue DP-SGD will often converge to a neighborhood of a different point.

Second, we argue that DP-SGD introduces bias. This was already pointed out in Chen et al. (2020)'s examples 1 and 2. In Section C.5 we also showed experiments demonstrating this phenomenon. Below, we provide a simple example which we can handle (almost) analytically.

A simple situation where bias occurs and DP-SGD does not converge to an optimum of F is when errors aren't symmetrically distributed, e.g., positive errors are less frequent but larger than negative errors.

Consider the scenario of simple linear regression. A common assumption of linear regression is that instances are of the form (x_i, y_i) where x_i is drawn from some distribution P_x and $y_i = ax_i + b + e_i$ where e_i is drawn from some zero-mean distribution P_e. When no other evidence is available, one often assume P_e to be Gaussian, but this is not necessarily the case. Suppose for our example that P_x is the uniform distribution over $[0, 1]$ and P_e only has two possible values, in particular $P_e(9) = 0.1$, $P_e(-1) = 0.9$ and $P_e(e) = 0$ for $e \notin \{9, -1\}$. So with high probability there is a small negative error e_i while with small probability there is a large positive error, while the average e_i is still 0. Consider a dataset $Z = \{(x_i, y_i)\}_{i=1}^n$. Let us consider a model $f(x) = \theta_1 x\theta_2$ and let us use the square loss $\mathcal{L}(\theta, Z) = \sum_{i=1}^n \ell(x_i, y_i)/n$ with $\ell(\theta, x, y) = (\theta_1 x + \theta_2 - y)^2$. Then, the gradient is

$$\nabla_\theta \ell(\theta, x, y) = (2(\theta_1 x + \theta_2 - y)x, 2(\theta_1 x + \theta_2 - y))$$

For an instance (x_i, y_i) with $y_i = ax_i + b + e_i$, this implies

$$\nabla_\theta \ell(\theta, x_i, y_i) = (2((\theta_1 - a)x_i + (\theta_2 - b) - e_i)x_i, 2((\theta_1 - a)x_i + (\theta_2 - b) - e_i))$$

For sufficiently large datasets Z where empirical loss approximates population loss, the gradient considered by Lip-DP-SGD will approximate

$$\begin{aligned}
\nabla_\theta \mathcal{L}(\theta, Z) &\approx \sum_{e\in\{10,\}} P_e(e) \int_0^1 \nabla_\theta \ell(\theta, x, ax + b + e)\mathrm{d}x \\
&= \sum_{e\in\{10,\}} P_e(e) \int_0^1 (2((\theta_1 - a)x + (\theta_2 - b) - e)x, 2((\theta_1 - a)x + (\theta_2 - b) - e))\,\mathrm{d}x \\
&= \int_0^1 \big(2((\theta_1 - a)x^2 + (\theta_2 - b)x - x\mathbb{E}[e]), 2((\theta_1 - a)x + (\theta_2 - b) - \mathbb{E}[e])\big)\,\mathrm{d}x \\
&= (2((\theta_1 - a)/3 + (\theta_2 - b)/2), 2((\theta_1 - a)/2 + (\theta_2 - b)))
\end{aligned}$$

This gradient becomes zero if $\theta_1 = a$ and $\theta_2 = b$ as intended.

However, if we use gradient clipping with threshold $C = 1$ as in DP-SGD, we get:

$$\begin{aligned}
\tilde{g} &\approx \sum_{e\in\{10,\}} P_e(e) \int_0^1 clip_1\left(\nabla_\theta \ell(\theta, x, ax + b + e)\right)\mathrm{d}x \\
&= \sum_{e\in\{10,\}} P_e(e) \int_0^1 clip_1\left((2((\theta_1 - a)x + (\theta_2 - b) - e)x, 2((\theta_1 - a)x + (\theta_2 - b) - e))\right)\mathrm{d}x
\end{aligned}$$

While for a given e for part of the population $(\theta_1 - a)x + \theta_2 - b$ may be small, for a fraction of the instances the gradients are clipped. For the instances with $e = 9$ this effect is stronger. The result is that for $\theta_1 = a$ and $\theta_2 = b$ the average clipped gradient $\tilde{g}$ doesn't become zero anymore, in particular $\|\tilde{g}\| = 0.7791$. In fact, $\tilde{g}$ becomes zero for $\theta_1 = a + 0.01765$ and $\theta_2 = b + 0.94221$. Figure E illustrates this situation.

Figure 5: An example of gradient clipping causing bias, here the average gradient becomes zero at $(0, 0)$ while the average clipped gradient is 0 at another point, causing convergence of DP-SGD to that point rather than the correct one.