



**HAL**  
open science

## DP-SGD with weight clipping

Antoine Barczewski, Jan Ramon

► **To cite this version:**

Antoine Barczewski, Jan Ramon. DP-SGD with weight clipping. CAp (Conférence sur l'Apprentissage automatique) 2024, SSFAM (Société Savante Française d'Apprentissage Machine); AFRIF (Association Française pour la Reconnaissance et l'Interprétation des Formes), Jul 2024, Lille (France), France. 10.48550/arXiv.2310.18001 . hal-04614505

**HAL Id: hal-04614505**

**<https://hal.science/hal-04614505v1>**

Submitted on 17 Jun 2024

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# DP-SGD with weight clipping

Antoine Barczewski<sup>1</sup> and Jan Ramon<sup>2</sup>

<sup>1</sup>Université Lille, Inria

<sup>2</sup>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

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

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

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

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

## 60 2 Preliminaries and background

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

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

### 65 2.1 Differential Privacy

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

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

70 **Definition 2 (differential privacy Dwork and Roth (2013)).** *Let  $\epsilon > 0$  and  $\delta > 0$ . Let  
71  $\mathcal{A} : \mathcal{Z}^* \rightarrow \mathcal{O}$  be a randomized algorithm taking as input datasets from  $\mathcal{Z}^*$ . The algorithm  $\mathcal{A}$  is  
72  $(\epsilon, \delta)$ -differentially private ( $(\epsilon, \delta)$ -DP) if for every pair of adjacent datasets  $(Z_1, Z_2) \in \mathcal{Z}_{\sim}^*$ , and for  
73 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  
74 say that  $\mathcal{A}$  is  $\epsilon$ -DP.*

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

**Definition 3 (sensitivity).** The  $\ell_2$ -sensitivity of a function  $f : \mathcal{Z} \rightarrow \mathbb{R}^p$  is

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

**Lemma 4 (Gaussian mechanism).** Let  $f : \mathcal{Z} \rightarrow \mathbb{R}^p$  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  $(\epsilon, \delta)$ -DP.

## 2.2 Empirical risk minimization

Unless made explicit otherwise we will consider databases  $Z = \{z_i\}_{i=1}^n$  containing  $n$  instances  $z_i = (x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$  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  $f_\theta : \mathcal{X} \rightarrow \hat{\mathcal{Y}}$  (with  $\hat{\mathcal{Y}} \subseteq \mathbb{R}$ ) parameterized by  $\theta \in \Theta \subseteq \mathbb{R}^p$ , so it minimizes the expected loss  $\mathcal{L}(\theta) = \mathbb{E}_z[\mathcal{L}(\theta; z)]$ , where  $\mathcal{L}(\theta; z) = \ell(f_\theta(x), y)$  is the loss of the model  $f_\theta$  on data point  $z$ . One can approximate  $\mathcal{L}(\theta)$  by

$$\hat{R}(\theta; Z) = \frac{1}{n} \sum_{i=1}^n \mathcal{L}(\theta; z_i) = \frac{1}{n} \sum_{i=1}^n \ell(f_\theta(x_i), y_i),$$

the empirical risk of model  $f_\theta$ . 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  $f_\theta : \mathbb{R}^n \rightarrow \mathbb{R}^m$  is a function which can be expressed as

$$f_\theta = f_{\theta_K}^{(K)} \circ \dots \circ f_{\theta_1}^{(1)}$$

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

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

106 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

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

## 107 2.4 Regularization

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

115 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  $\Gamma_k = \{\Gamma_{k,1} \dots \Gamma_{k,|\Gamma_k|}\}$  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:

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

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

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

117 with  $\kappa$  a small constant.

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

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

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

Note that  $\theta_k$  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

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

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

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

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  $\max_{x_k} \left\| \frac{\partial \mathcal{L}_k}{\partial \theta_k} \right\|_2$ , 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  $\Delta_k$  if the layers  $f_\theta^{(k)}$  and the loss  $\ell$  are Lipschitz. If so, since all  $f_\theta^{(k)}$  and  $\ell$  are differentiable on any  $x_k$ , per Rademacher's theorem Rademacher (1919),  $\left\| \frac{\partial \mathcal{L}_k}{\partial x_k} \right\|_2$  is bounded by the Lipschitz value of  $\mathcal{L}_k$ . We only need to find a tight upper bound of this Lipschitz value.

### 136 3.2 Estimating lipschitz values

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

139 **Loss function and activation layer.** Examples of Lipschitz losses encompass Softmax Cross-  
140 entropy, Cosine Similarity, and Multiclass Hinge. When it comes to activation layers, layers composed  
141 of an activation function, several prevalent ones, such as ReLU, tanh, and Sigmoid are 1-Lipschitz. We  
142 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):

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

with  $\alpha$  an hyperparameter. It is easy to see that the sensitivity is bounded by

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

143 Note that a group normalization layer has no trainable parameters.

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

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

**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

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

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

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

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

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

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

---

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

---

```

1: Input:  $K$  layer feed-forward model  $f$ , parameters  $\theta$ , parameter norm  $u^{(\theta)}$ , max input norm  $X_1$ ,
   upper bound of loss Lipschitz value  $l_{K+1}$ .
2: for  $k = 1$  to  $K$  do ▷ Forward pass
3:   if  $\theta_k = \emptyset$  then
4:     if  $k$ -th layer is a normalization layer then
5:        $X_{k+1} \leftarrow \min(|x_{k+1}|^{\frac{1}{2}}, \frac{X_k}{\alpha})$ 
6:     else ▷ e.g., activation layer
7:        $X_{k+1} \leftarrow X_k L_{x_k}$ 
8:     end if
9:   else
10:     $X_{k+1} \leftarrow X_k u_k^{(\theta)}$ 
11:   end if
12: end for
13: for  $k = K$  to  $1$  do ▷ Backward pass
14:    $l_k \leftarrow l_{k+1} L_{x_k}$ 
15:    $\Delta_k \leftarrow l_{k+1} L_{\theta_k}$ 
16: end for
17: Output:  $(\Delta_1, \dots, \Delta_{K-1}, \Delta_K)$ .
```

---

### 158 3.3 Lip-DP-SGD

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

162 **Theorem 7.** *Given a feed-forward model  $f_\theta$  composed of Lipschitz constrained operators, a Lipschitz*  
 163 *loss  $\ell$  and a bounded input norm  $X_1$ , LIP-DP-SGD is differentially private.*

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



---

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

---

```

1: Input: Data set  $Z \in \mathcal{Z}^*$ , feed-forward model  $f_\theta$ , loss function  $\mathcal{L}$ , hypothesis space  $\Theta \subseteq \mathbb{R}^k$ , number
   of epochs  $T$ , noise multiplier  $\sigma$ , batch size  $s \geq 1$ , learning rate  $\eta$ , max gradient norm  $C$ 
2: Initialize  $\tilde{\theta}$  randomly from  $\Theta$ 
3:  $(u^{(\theta)}, \tilde{\theta}) \leftarrow \text{CLIPWEIGHTS}(\tilde{\theta}, C)$ 
4: for  $t \in [T]$  do
5:    $(\Delta_k)_{k=1}^K \leftarrow \text{LAYERSENSITIVITY}(f, \tilde{\theta}, u^{(\theta)})$ 
6:    $V \leftarrow \emptyset$  ▷ Poisson sampling
7:   while  $V = \emptyset$  do
8:     for  $z \in Z$  do
9:       With probability  $s/|Z|$ :  $V \leftarrow V \cup \{z\}$ 
10:    end for
11:  end while
12:  for  $k = 1 \dots K$  do ▷ gradient per layer
13:    Draw  $b_k \sim \mathcal{N}(0, \sigma^2 \Delta_k^2 \mathbb{I})$ 
14:     $\tilde{g}_k \leftarrow \frac{1}{|V|} \left( \sum_{i=1}^{|V|} \nabla_{\tilde{\theta}_k} \ell(f_{\tilde{\theta}}(x_i), y_i) + b_k \right)$ 
15:     $\tilde{\theta}_k \leftarrow \tilde{\theta}_k - \eta(t) \tilde{g}_k$ 
16:  end for
17:   $(u^{(\theta)}, \tilde{\theta}) \leftarrow \text{CLIPWEIGHTS}(\tilde{\theta}, C)$ 
18: end for
19: Output:  $\tilde{\theta}$  and compute  $(\epsilon, \delta)$  with privacy accountant.
20: function CLIPWEIGHTS( $\tilde{\theta}, C$ )
21:   for  $k = 1 \dots K$  do
22:     if  $\tilde{\theta}_k \neq \emptyset$  then
23:        $u_k^{(\tilde{\theta})} \leftarrow \min(C, \|\tilde{\theta}_k\|)$ 
24:        $\tilde{\theta}_k \leftarrow u_k^{(\tilde{\theta})} \tilde{\theta}_k / \|\tilde{\theta}_k\|$ 
25:     end if
26:   end for
27:   return  $(u^{(\theta)}, \tilde{\theta})$ 
28: end function

```

---

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

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

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

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

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

### 191 3.4 Avoiding the bias of gradient clipping

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

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

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

201 Furthermore, Chen et al. (2020) shows an example showing that gradient clipping can introduce  
202 bias. We add a more detailed discussion in Appendix E. Hence, DP-SGD does not necessarily converge  
203 to a local optimum of  $F(\theta, Z)$ , even when sufficient data is available to estimate  $\theta$ . While Lip-DP-SGD  
204 can only find models in  $\Theta_{\leq C}$  and this may introduce another suboptimality, as our experiments will  
205 show this is only a minor drawback in practice, while also others observed that Lipschitz networks have  
206 good properties Béthune et al. (2023). Moreover, it is easy to check whether Lip-DP-SGD outputs  
207 parameters on the boundary of  $\Theta_{\leq C}$  and hence the model could potentially improve by relaxing the  
208 weight norm constraint. In contrast, it may not be feasible to detect that DP-SGD is outputting  
209 potentially suboptimal parameters. Indeed, consider a federated learning setting (e.g., Bonawitz et al.  
210 (2017)) where data owners collaborate to compute a model without revealing their data. Each data  
211 owner locally computes a gradient and clips it, and then the data owners securely aggregate their  
212 gradients and send the average gradient to a central party updating the model. In such setting, for  
213 privacy reasons no party would be able to evaluate that gradient clipping introduces a strong bias in  
214 some direction. Still, our experiments show that in practice at the time of convergence for the best  
215 hyperparameter values clipping is still active for a significant fraction of gradients (See Appendix C.5)

## 216 4 Experimental results

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

### 218 4.1 Experimental setup

219 We consider the following experimental questions:

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

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

226 **Implementation.** We implemented both the DP-SGD and Lip-DP-SGD methods to ensure that  
 227 comparisons were made under consistent model structures and preprocessing conditions.

228 To answer question Q2, we also implemented Fix-Lip-DP-SGD, a version of Lip-DP-SGD limited to  
 229 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  
 230 Algorithm 2.

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

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

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

244 **Infrastructure.** All experiments were orchestrated across dual Tesla P100 GPU platforms (12GB  
 245 capacity), operating under CUDA version 10, with a 62GB RAM provision for Fashion-MNIST and  
 246 CIFAR-10. Remaining experiments were performed on an E5-2696V2 Processor setup, equipped with  
 247 8 vCPUs and a 52GB RAM cache. The total runtime of the experiments was approximately 50 hours,  
 248 which corresponds to an estimated carbon emission of 1.96 kg Lacoste et al. (2019). More details on  
 249 the experimental setup and an analysis of the runtime can be found in Appendix C.

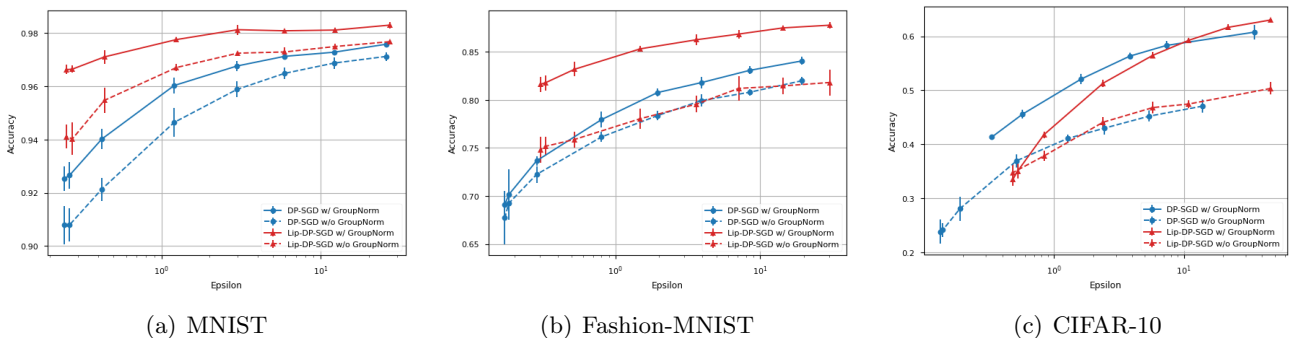


Figure 1: Accuracy results, with a fixed  $\delta = 10^{-5}$ , for the MNIST (1(a)), Fashion-MNIST (1(b)), and CIFAR-10 (1(c)) datasets, comparing Lip-DP-SGD (in red) and DP-SGD (in blue), lines are dashed when the underlying model is implemented without group normalization. Vertical lines represent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

Table 1: Accuracy and  $\epsilon$  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 $n \times \#$ features $p$ )	$\epsilon$	DP-SGD	Lip-DP-SGD	Fix-Lip-DP-SGD
Adult Income (48842x14) Becker et al. (1996)	0.414	0.824	<b>0.831</b>	0.804
Android (29333x87) Mathur et al. (2022)	1.273	0.951	<b>0.959</b>	0.945
Breast Cancer (569x32) Wolberg et al. (1995)	1.672	0.773	<b>0.798</b>	0.7
Default Credit (30000x24) Yeh (2016)	1.442	0.809	<b>0.816</b>	0.792
Dropout (4424x36) Realinho et al. (2021)	1.326	0.763	<b>0.819</b>	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	<b>0.931</b>	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  $\theta$  to a fixed value but to exploit cases where it becomes smaller.

## 5 Related Work

**DP-SGD.** DP-SGD algorithms have been developed 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

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

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

## 295 6 Conclusion and discussion

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

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

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

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

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

## 317 Acknowledgements

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

## 320 References

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

- 324 Andrew, G., Thakkar, O., McMahan, H. B., and Ramaswamy, S. (2022). Differentially Private  
325 Learning with Adaptive Clipping. In *Advances in Neural Information Processing Systems*. arXiv.  
326 arXiv:1905.03871 [cs, stat].
- 327 Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In  
328 *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML’17, page  
329 214–223. JMLR.org.
- 330 Balandat, M., Karrer, B., Jiang, D. R., Daulton, S., Letham, B., Wilson, A. G., and Bakshy, E. (2020).  
331 BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. arXiv:1910.06403 [cs,  
332 math, stat].
- 333 Bassily, R., Feldman, V., Talwar, K., and Guha Thakurta, A. (2019). Private Stochastic Convex  
334 Optimization with Optimal Rates. In *Advances in Neural Information Processing Systems*, volume 32.  
335 Curran Associates, Inc.
- 336 Bassily, R., Smith, A., and Thakurta, A. (2014). Private empirical risk minimization: Efficient  
337 algorithms and tight error bounds. In *2014 IEEE 55th Annual Symposium on Foundations of  
338 Computer Science*, pages 464–473.
- 339 Becker, Barry, Kohavi, and Ronny (1996). Adult. UCI Machine Learning Repository. DOI: 10.24432/  
340 C5XW20.
- 341 Bethune, L., Massena, T., Boissin, T., Prudent, Y., Friedrich, C., Mamalet, F., Bellet, A., Serrurier,  
342 M., and Vigouroux, D. (2023). Dp-sgd without clipping: The lipschitz neural network way.
- 343 Béthune, L., Novello, P., Coiffier, G., Boissin, T., Serrurier, M., Vincenot, Q., and Troya-Galvis, A.  
344 (2023). Robust one-class classification with signed distance function using 1-lipschitz neural networks.  
345 In *Proceedings of the 40th International Conference on Machine Learning*, ICML’23. JMLR.org.
- 346 Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal,  
347 A., and Seth, K. (2017). Practical secure aggregation for privacy-preserving machine learning. In  
348 *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS  
349 ’17, page 1175–1191, New York, NY, USA. Association for Computing Machinery.
- 350 Chaudhuri, K., Monteleoni, C., and Sarwate, A. D. (2011). Differentially Private Empirical Risk  
351 Minimization. *Journal of Machine Learning Research*, page 41.
- 352 Chen, C. and Lee, J. (2020). Stochastic adaptive line search for differentially private optimization. In  
353 *2020 IEEE International Conference on Big Data (Big Data)*, pages 1011–1020.
- 354 Chen, X., Wu, S. Z., and Hong, M. (2020). Understanding gradient clipping in private sgd: A geometric  
355 perspective. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances  
356 in Neural Information Processing Systems*, volume 33, pages 13773–13782. Curran Associates, Inc.
- 357 Daulton, S., Balandat, M., and Bakshy, E. (2020). Differentiable Expected Hypervolume Improvement  
358 for Parallel Multi-Objective Bayesian Optimization. arXiv:2006.05078 [cs, math, stat].
- 359 De, S., Berrada, L., Hayes, J., Smith, S. L., and Balle, B. (2022). Unlocking High-Accuracy Differentially  
360 Private Image Classification through Scale. *arXiv preprint arXiv:2204.13650*.
- 361 Deng, L. (2012). The mnist database of handwritten digit images for machine learning research. *IEEE  
362 Signal Processing Magazine*, 29(6):141–142.

- 363 Dwork, C., McSherry, F., Nissim, K., and Smith, A. (2006). Calibrating Noise to Sensitivity in Private  
364 Data Analysis. In Halevi, S. and Rabin, T., editors, *Theory of Cryptography*, pages 265–284, Berlin,  
365 Heidelberg. Springer Berlin Heidelberg.
- 366 Dwork, C. and Roth, A. (2013). The Algorithmic Foundations of Differential Privacy. *Foundations  
367 and Trends® in Theoretical Computer Science*, 9(3-4):211–407.
- 368 Dwork, C., Rothblum, G. N., and Vadhan, S. (2010). Boosting and Differential Privacy. In *2010 IEEE  
369 51st Annual Symposium on Foundations of Computer Science*, pages 51–60, Las Vegas, NV, USA.  
370 IEEE.
- 371 Gouk, H., Frank, E., Pfahringer, B., and Cree, M. J. (2020). Regularisation of neural networks by  
372 enforcing lipschitz continuity.
- 373 Hofmann, H. (1994). Statlog (German Credit Data). UCI Machine Learning Repository. DOI:  
374 10.24432/C5NC77.
- 375 Ioffe, S. and Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by  
376 Reducing Internal Covariate Shift. *ICML*.
- 377 Kasiviswanathan, S. P., Lee, H. K., Nissim, K., Raskhodnikova, S., and Smith, A. (2011). What can  
378 we learn privately? *SIAM Journal on Computing*, 40(3):793–826.
- 379 Kifer, D., Smith, A., and Thakurta, A. (2012). Private Convex Empirical Risk Minimization and  
380 High-dimensional Regression. In *Proceedings of the 25th Annual Conference on Learning Theory*,  
381 pages 25.1–25.40. JMLR Workshop and Conference Proceedings. ISSN: 1938-7228.
- 382 Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *International Conference  
383 on Learning Representations*.
- 384 Koskela, A. and Honkela, A. (2020). Learning Rate Adaptation for Differentially Private Learning. In  
385 *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*,  
386 pages 2465–2475. PMLR. ISSN: 2640-3498.
- 387 Krizhevsky, A., Nair, V., and Hinton, G. (2009). Cifar-10 (canadian institute for advanced research).  
388 URL <http://www.cs.toronto.edu/kriz/cifar.html>.
- 389 Lacoste, A., Luccioni, A., Schmidt, V., and Dandres, T. (2019). Quantifying the carbon emissions of  
390 machine learning.
- 391 Lee, J. and Kifer, D. (2018). Concentrated differentially private gradient descent with adaptive  
392 per-iteration privacy budget. In *Proceedings of the 24th ACM SIGKDD International Conference on  
393 Knowledge Discovery & Data Mining, KDD '18*, page 1656–1665, New York, NY, USA. Association  
394 for Computing Machinery.
- 395 Mathur, Akshay, Mathur, and Akshay (2022). NATICUSdroid (Android Permissions) Dataset. UCI  
396 Machine Learning Repository.
- 397 McMahan, B., Moore, E., Ramage, D., Hampson, S., and Arcas, B. A. y. (2017). Communication-  
398 Efficient Learning of Deep Networks from Decentralized Data. In Singh, A. and Zhu, J., editors,  
399 *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54  
400 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR.
- 401 Mironov, I., Talwar, K., and Zhang, L. (2019). Rényi Differential Privacy of the Sampled Gaussian  
402 Mechanism. *arXiv e-prints*, page arXiv:1908.10530.

- 403 Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. (2018). Spectral normalization for generative  
404 adversarial networks. In *Advances in Neural Information Processing Systems*.
- 405 Papernot, N. and Steinke, T. (2022). Hyperparameter Tuning with Renyi Differential Privacy. In  
406 *International Conference on Learning Representations*. arXiv. arXiv:2110.03620 [cs].
- 407 Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein,  
408 N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy,  
409 S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-  
410 performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F.,  
411 Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages  
412 8024–8035. Curran Associates, Inc.
- 413 Rademacher, H. (1919). Über partielle und totale differenzierbarkeit von funktionen mehrerer variabeln  
414 und über die transformation der doppelintegrale. *Mathematische Annalen*, 79:340–359.
- 415 Rajkovic, V. (1997). Nursery. UCI Machine Learning Repository. DOI: 10.24432/C5P88W.
- 416 Realinho, V., Vieira Martins, M., Machado, J., and Baptista, L. (2021). Predict students' dropout and  
417 academic success. UCI Machine Learning Repository. DOI: 1.
- 418 Scaman, K. and Virmaux, A. (2019). Lipschitz regularity of deep neural networks: analysis and efficient  
419 estimation. In *Advances in Neural Information Processing Systems*. arXiv. arXiv:1805.10965 [cs,  
420 stat].
- 421 Song, S., Steinke, T., Thakkar, O., and Thakurta, A. G. (2020). Evading the curse of dimensionality  
422 in unconstrained private generalized linear problems. In *Proceedings of the 23th International  
423 Conference on Artificial Intelligence and Statistics*.
- 424 von Mises, R. and Pollaczek-Geiringer, H. (1929). Praktische verfahren der gleichungsauffösung .  
425 *Zamm-zeitschrift Fur Angewandte Mathematik Und Mechanik*, 9:58–77.
- 426 Wolberg, William, Mangasarian, Olvi, Street, Nick, and Street, W. (1995). Breast Cancer Wisconsin  
427 (Diagnostic). UCI Machine Learning Repository. DOI: 10.24432/C5DW2B.
- 428 Wu, Y. and He, K. (2018). Group normalization. *International Journal of Computer Vision*, 128:742 –  
429 755.
- 430 Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-MNIST: a Novel Image Dataset for Benchmarking  
431 Machine Learning Algorithms. arXiv:1708.07747 [cs, stat].
- 432 Yeh, I.-C. (2016). default of credit card clients. UCI Machine Learning Repository. DOI: 10.24432/  
433 C55S3H.
- 434 Yoshida, Y. and Miyato, T. (2017). Spectral Norm Regularization for Improving the Generalizability  
435 of Deep Learning. arXiv:1705.10941 [cs, stat].
- 436 Yu, L., Liu, L., Pu, C., Gursoy, M. E., and Truex, S. (2019). Differentially Private Model Publishing  
437 for Deep Learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 332–349.  
438 arXiv:1904.02200 [cs].
- 439 Ziller, A., Usynin, D., Knolle, M., Prakash, K., Trask, A., Braren, R., Makowski, M., Rueckert, D.,  
440 and Kaissis, G. (2021). Sensitivity analysis in differentially private machine learning using hybrid  
441 automatic differentiation. arXiv:2107.04265 [cs].



## 442 A Gradient clipping based DP-SGD

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

---

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

---

**Input:** Data set  $Z \in \mathcal{Z}^*$ , model  $f_\theta$ , loss function  $\mathcal{L}$ , hypothesis space  $\Theta \subseteq \mathbb{R}^k$ , privacy parameters  $\epsilon$  and  $\delta$ , noise multiplier  $\sigma$ , batch size  $s \geq 1$ , learning rate  $\eta$ , max gradient norm  $C$

Initialize  $\tilde{\theta}$  randomly from  $\Theta$

**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  $(\epsilon, \delta)$  with privacy accountant.

---

## 445 B Estimating Lipschitz values

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

Layer	Definition	Lip. value on input $x_k$	Lip. value on parameter $\theta_k$
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	$y \log \text{SOFTMAX}(x_{K+1})/\tau$	$\sqrt{2}/\tau$	-
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.

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

## 452 B.1 Details for the convolutional layer

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

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

453 *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:

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

454 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,

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

455

□

## 456 C Experiments

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

462 **C.1 Hyperparameters**

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

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

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

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

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

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

Hyperparameter	Range
Noise multiplier $\sigma$	[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]
$\alpha$ (group normalization)	$[0.1/( x_k^{(k:q)} ), 1/( x_k^{(k:q)} )]$

Table 3: Summary of hyperparameter space.

489 **C.2 Models**

490 Table 4 shows details of the models we used to train on tabular and image datasets. We consider 7  
 491 tabular datasets: adult income Becker et al. (1996), android permissions Mathur et al. (2022), breast  
 492 cancer Wolberg et al. (1995), default credit Yeh (2016), dropout Realinho et al. (2021), German credit  
 493 Hofmann (1994) and nursery Rajkovic (1997). See Table 1 for the number of instances and features for  
 494 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.

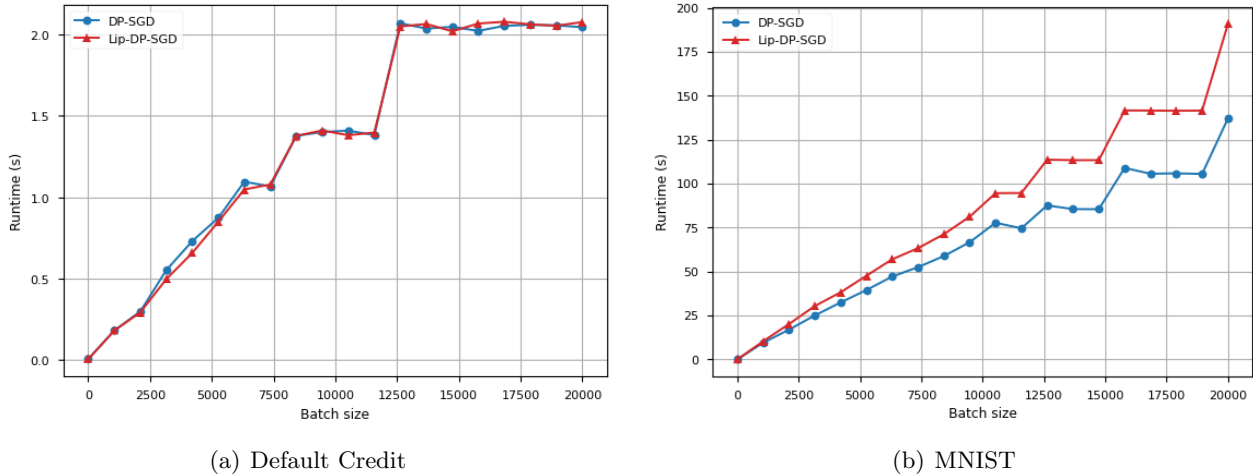


Figure 2: Mean runtime in seconds per batch size on one epoch over the Default Credit dataset 2(a) and the MNIST dataset 2(b) comparing DP-SGD (in blue) and Lip-DP-SGD (in red).

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

## 511 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

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

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

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

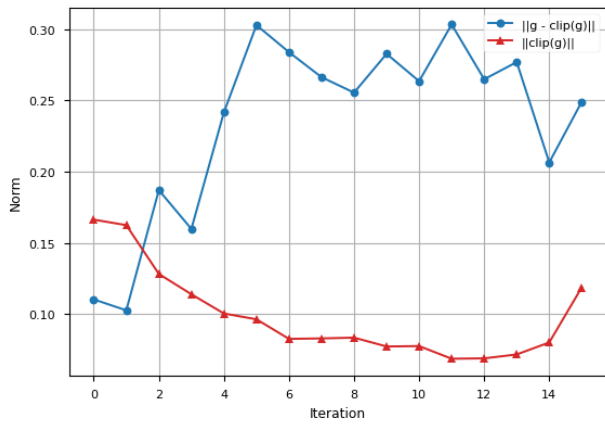
515 One can observe for both considered datasets that while the model converges and the average  
516 clipped gradient decreases, the error between DP-SGD’s average clipped gradient and the true average  
517 gradient increases. At the end, the error in the gradient caused by clipping is significant,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

## 519 C.6 Regularization techniques.

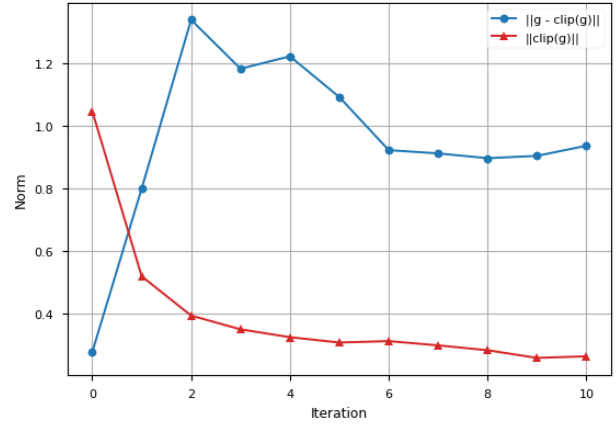
520 In De et al. (2022) multiple other regularization/optimization techniques are proposed. In our  
521 experiments, we found that while these techniques sometimes help, there is no significant or systematic  
522 difference in the effect on the results of DP-SGD and Lip-DP-SGD. For example, Figure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

## 527 D Lip-DP-SGD library

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

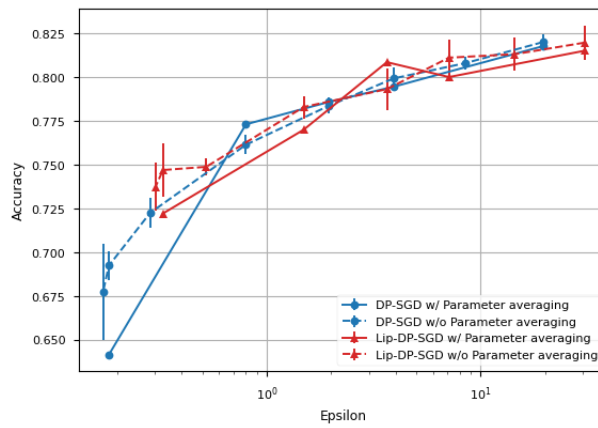


(a) Dropout



(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.

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

534 ‘README.md’, provided in the supplementary materials, details how to run the library and how  
 535 to reproduce the experiments.

## 536 E Avoiding the bias of gradient clipping

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

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

547 First, let us consider convergence.

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

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

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

We introduce a slack variable  $\zeta$ :

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

Using Lagrange multipliers, we should minimize

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

552 An optimum in  $\theta, \lambda$  and  $\zeta$  satisfies

$$\nabla_\theta F(\theta, Z) - \lambda\theta = 0 \tag{11}$$

$$\|\theta\|_2 + \zeta^2 - C = 0 \tag{12}$$

$$2\lambda\zeta = 0 \tag{13}$$

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

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

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

570 Second, we argue that DP-SGD introduces bias. This was already pointed out in Chen et al.  
 571 (2020)'s examples 1 and 2. In Section C.5 we also showed experiments demonstrating this phenomenon.  
 572 Below, we provide a simple example which we can handle (almost) analytically.

573 A simple situation where bias occurs and DP-SGD does not converge to an optimum of  $F$  is when  
 574 errors aren't symmetrically distributed, e.g., positive errors are less frequent but larger than negative  
 575 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

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

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

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

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

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

$$\begin{aligned} \tilde{g} &\approx \sum_{e \in \{10, \}} P_e(e) \int_0^1 \text{clip}_1(\nabla_{\theta} \ell(\theta, x, ax + b + e)) dx \\ &= \sum_{e \in \{10, \}} P_e(e) \int_0^1 \text{clip}_1((2((\theta_1 - a)x + (\theta_2 - b) - e)x, 2((\theta_1 - a)x + (\theta_2 - b) - e))) dx \end{aligned}$$



580 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  
581 instances the gradients are clipped. For the instances with  $e = 9$  this effect is stronger. The result is  
582 that for  $\theta_1 = a$  and  $\theta_2 = b$  the average clipped gradient  $\tilde{g}$  doesn't become zero anymore, in particular  
583  $\|\tilde{g}\| = 0.7791$ . In fact,  $\tilde{g}$  becomes zero for  $\theta_1 = a + 0.01765$  and  $\theta_2 = b + 0.94221$ . Figure E illustrates  
584 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.

