Wenxiao Wang, Tianhao Wang, Lun Wang, Nanqing Luo, Pan Zhou, Dawn Song, and Ruoxi Jia

DPlis: Boosting Utility of Differentially Private Deep Learning via Randomized Smoothing

Abstract: Deep learning techniques have achieved remarkable performance in wide-ranging tasks. However, when trained on privacy-sensitive datasets, the model parameters may expose private information in training data. Prior attempts for differentially private training, although offering rigorous privacy guarantees, lead to much lower model performance than the non-private ones. Besides, different runs of the same training algorithm produce models with large performance variance. To address these issues, we propose DPlis–Differentially Private Learning wIth Smoothing. The core idea of DPlis is to construct a smooth loss function that favors noise-resilient models lying in large flat regions of the loss landscape. We provide theoretical justification for the utility improvements of DPlis. Extensive experiments also demonstrate that DPlis can effectively boost model quality and training stability under a given privacy budget.

Keywords: differential privacy, stochastic gradient descent, deep learning, randomized smoothing

DOI 10.2478/popets-2021-0065
Received 2021-02-28; revised 2021-06-15; accepted 2021-06-16.

1 Introduction

Recent advances in deep neural networks (DNNs) has led to state-of-the-art performances in a wide variety of tasks, including, among others, image recognition and natural language processing. The availability of large datasets is indispensable for these advances. However, in many application domains of deep learning, such as healthcare, finance, and location-based services, the data may contain privacy-sensitive information. As DNNs tend to memorize training data [5, 16, 67], the sensitive information might be leaked with the release of the model [46, 49, 50].

Differential privacy (DP), a canonical privacy notion that provides provable privacy guarantees, has been applied to mitigate training data leakage. DP aims at hiding the presence of every individual record from the output of an algorithm performed on private data. To achieve this, DP characterizes the sensitivity of output to the change of one record in an arbitrary input dataset and further adds noise to the output that is proportional to the sensitivity. The characterization of sensitivity is challenging for DNNs, as the dependency of the large size parameters on the training data is difficult to understand and trace.

Prior work presents two general threads of ideas to deal with the challenge of sensitivity characterization in deep learning. One is to recognize that DNNs are mostly trained iteratively via stochastic gradient descent (SGD) methods and thus one can bound the sensitivity of each SGD step by clipping gradient norm and then perturb the gradients accordingly. By ensuring that each SGD step is differentially private, the final output model satisfies a certain level of DP given the composition theorem. As training DNNs involves a large number of iterations, it is necessary to tightly track cumulative privacy loss during training and halt when the loss hits the privacy budget. Moments accountant [1, 61] provides a much tighter composition analysis for Gaussian noise applied to subsampled data, compared to the standard advanced composition theorem [14] in DP. The combination of the noisy SGD and moment accountants is often referred to as DP-SGD and has been widely used for building differentially private DNNs. However, despite DP-SGD providing a principled, easily implementable framework for building differentially private DNNs, the *prediction accuracy* of the privately trained models is severely impaired due to the large noise added into each SGD step. Another side-effect of DP-SGD of-

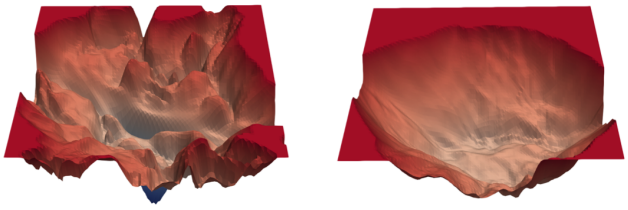
Wenxiao Wang: Tsinghua University.
E-mail: wangwx20@mails.tsinghua.edu.cn
Tianhao Wang: Harvard University.
E-mail: tianhaowang@fas.harvard.edu
Lun Wang: University of California, Berkeley.
E-mail: wanglun@berkeley.edu
Nanqing Luo: Huazhong University of Science and Technology. E-mail: u201714868@hust.edu.cn
Pan Zhou: Huazhong University of Science and Technology.
E-mail: panzhou@hust.edu.cn
Dawn Song: University of California, Berkeley.
E-mail: dawnsong@cs.berkeley.edu
Ruoxi Jia: Virginia Tech. E-mail: ruoxijia@vt.edu

ten overlooked by the previous work is the lack of *stability*–different runs of the same algorithm results in drastically different models in terms of prediction accuracy.

Another thread of ideas to overcome the difficulty of sensitivity characterization in DNNs is based on Private Aggregation of Teacher Ensembles (PATE) [41, 42]. PATE first trains an ensemble of models (i.e., teacher models) on private data and then aggregates the predictions from different teacher models to label public data in a differentially private manner. Since post-processing differentially private results will not affect the privacy guarantee, a differentially private model can be directly constructed via training on the labeled public data. Overall, PATE converts the problem of characterizing the sensitivity of a training algorithm to that of a label aggregation function, which is much easier to analyze. However, PATE requires access to a large public dataset similar to private data and this prerequisite cannot always be satisfied in practice.

Hence, in this paper, we focus on DP-SGD and aim at boosting the utility of DP-SGD in terms of both prediction accuracy and stability. Existing improvement strategies for DP-SGD are based on the idea of modifying the design of *optimization algorithms* and *model architectures*, such as post-processing the noisy gradients in a way that can reduce noise variance [57], adjusting the gradient clipping threshold to the norms of the SGD updates [55], adopting a different error backpropagation method [31], and using a family of bounded activation functions [43]. In this paper, we propose an improvement strategy that is complementary to existing ones. The proposed strategy is motivated by the following question: Can we modify the *optimization objective* to make it more suitable for differentially private learning?

As the first step to answer the question, we examine the issues associated with standard optimization objective functions (i.e., learning loss functions) for DNNs. Figure 1a exemplifies the loss surface of a DNN, which is irregular and contains a lot of sharp local minima. Intuitively, such a loss landscape can have detrimental effects on the performance and stability of DP-SGD. Firstly, the noise injected by DP-SGD can result in significant loss increases near each sharp local minimum, which in turn leads to low prediction accuracy. Secondly, with this irregular landscape, noise perturbation along different directions could cause very different loss changes; thus, different runs of the same DP-SGD algorithm might still have large performance variance. Overall, the "bumpy" and irregular loss landscape associated with DNNs makes standard learning loss functions unsuitable for performing DP-SGD. Instead, loss functions with flat minima as in Figure 1b would be more desirable for DP-SGD because they are more noise-tolerant.



(a) Standard **(b)** Flat

Fig. 1. Loss surfaces of a ResNet-56 on 1% training data of CIFAR-10, without and with smoothing. The standard loss surface is irregular and contains many local minima, which is unfavorable given noise injected by DP-SGD. In comparison, a flat loss surface will be more noise-tolerant.

Contributions. Inspired by the observations and analysis above, we propose DPLIS, which improves the utility of DP-SGD by smoothing the learning loss function. Algorithmically, we leverage randomized smoothing to smooth the loss function, which convolves Gaussian noise with the original learning loss function. We theoretically connect smoothed loss with the stability and generalization of DP-SGD. Our experiments on diverse datasets and models show that DPLIS alone can often achieve $1 \sim 6\%$ prediction accuracy improvement on image data and $8 \sim 80\%$ perplexity reduction on language data. At the same time, DPLIS leads to more stable performance across different runs. Particularly, DPLIS is compatible with other DP-SGD improvement strategies that focus on modifying the optimization algorithm (e.g., [31, 55, 57]). Thus, DPLIS can potentially further boost the utility of differentially private DNNs when combined with other existing improvement strategies. Moreover, we show that the idea of smoothing learning loss can be easily integrated into PATE—a popular differentially private deep learning framework in the presence of public data. Our experimental results demonstrate that smoothing can usually improve the prediction accuracy by $1 \sim 4\%$.

As a side note, smoothing has been widely used in the machine learning community to improve the generalization of models [6, 63]. However, to the best of our knowledge, our work is the first to investigate smoothing in the context of privacy-preserving learning and provide theoretical bases for the utility gains of smoothing in DP-SGD.

Paper Organization. In Section 2, we include background information regarding Differential Privacy, Deep

Learning and their intersections. In Section 3, we motivate our idea with toy examples and present the design of our method, DPLIS. In Section 4, we derive theoretical results regarding the impact of DPLIS on stability and generalization of DP-SGD. In Section 5, we have the effectiveness of DPLIS verified empirically. We include a discussion of related work in Section 6. Finally we conclude our work in Section 7.

2 Background

2.1 Differential Privacy

Given a randomized function that takes a dataset or database as its input, DP [11–13] aims to hide the difference of output distribution between two neighboring inputs and provides provable privacy protection against adversaries of arbitrary computational power. The formal definition of DP is as follows.

Definition 1. ***(Differential Privacy)*** *A randomized function $\mathcal{F}$ gives (ε, δ)-differential privacy if for all pairs of adjacent datasets $D, D' \in \mathbb{D}$ that differ in at most one record and all $S \subseteq Range(\mathcal{F})$,*

$$\Pr[\mathcal{F}(D) \in S] \leq e^{\varepsilon} \Pr[\mathcal{F}(D') \in S] + \delta$$

A common practice to making an arbitrary function differentially private is through noising [11–13], which adds noise of a certain form to the output of the function, with the scale of noise proportional to the sensitivity of the function. The formal definition of sensitivity is as follows.

Definition 2. ***(Sensitivity)*** *The sensitivity Δ of a function f is:*

$$\Delta(f) = \max_{D \sim D'} \left\| f(D) - f(D') \right\|,$$

where $D \sim D'$ indicates that D and D' are two adjacent datasets.

The sensitivity captures the maximum change of the function outputs when an arbitrary input dataset changes by one entry. When we use ℓ_2-norm to measure the change, the corresponding sensitivity will be referred to as ℓ_2-sensitivity or $\Delta_2(f)$.

One prevalent way of noising in the context of differentially private deep learning [1, 57, 58, 66] is the Gaussian mechanism, which adds proper Gaussian noise to the function output based on the ℓ_2-sensitivity.

Definition 3. ***(Gaussian Mechanism)*** *The Gaussian mechanism with parameter σ, when applied to a function $f : \mathbb{D} \to \mathbb{R}^K$, adds zero-mean Gaussian noise with variance σ^2 in each of the K dimensions of $\mathcal{F}$'s output: $f(\cdot) + \mathcal{N}(0, \sigma^2 I)$.*

For $\sigma \geq \frac{\sqrt{2\ln(\frac{1.25}{\delta})}\Delta_2(f)}{\varepsilon}$ and $\varepsilon \in (0, 1)$, it is proven that the noised function above is $(\varepsilon, \delta)-$differentially private [13].

2.2 Deep Learning

A neural network f_θ is a composition of L parametric functions referred to as layers. Each layer consists of neurons, each of which provides one dimension of the layer's output. We denote the lth layer as f_{θ_l} and θ_l are the associated parameters that control the behavior of the layer. f_{θ_l} takes as its input the output of the previous layer $f_{\theta_{l-1}}$ and applies a nonlinear transformation to compute its own output. Given an input x, a neural network f_θ performs the following computation to predicts its label:

$$f_\theta(x) = f_{\theta_L} \circ \cdots \circ f_{\theta_1}(x)$$

where $\theta = [\theta_1, \ldots, \theta_L]$.

The parameter θ of a neural network is learned via solving an optimization problem. The optimization objective is the learning loss function $\mathcal{L}(f_\theta, D)$, which is the average of per-sample loss, i.e.

$$\mathcal{L}(f_\theta, D) = \frac{1}{|D|} \sum_{x \in D} \mathcal{L}(f_\theta, x)$$

where D is training data. With a slight abuse of notations, hereinafter, we will use $\mathcal{L}(\theta)$ as a shorthand notation for the average loss function $\mathcal{L}(f_\theta, D)$, and use $\mathcal{L}(\theta, x)$ to denote the sample-level loss function $\mathcal{L}(f_\theta, x)$.

The optimization problem is often solved via stochastic gradient descent (SGD) methods. At each step of SGD, a batch $B \subseteq D$ of samples is drawn from the training dataset D and the gradient of the average loss $\nabla_\theta \mathcal{L}(\theta) = \frac{1}{|D|} \sum_{x \in D} \nabla_\theta \mathcal{L}(\theta, x)$ will then be approximated by $\frac{1}{|B|} \sum_{x \in B} \nabla_\theta \mathcal{L}(\theta, x)$. With such an approximation, the following rule is applied iteratively to update the parameter:

$$\theta \leftarrow \theta - \eta \cdot \frac{1}{|B|} \sum_{x \in B} \nabla_\theta \mathcal{L}(\theta, x)$$

Algorithm 1 Differentially Private SGD

Require: examples $\{x_1, \ldots, x_N\}$, loss function $\mathcal{L}(\theta) = \frac{1}{N}\sum_i \mathcal{L}(\theta, x_i)$, learning rate η, noise multiplier σ, batch size L, clipping threshold C.

Initialize θ_0 randomly

for $t \in [T]$ **do**

Take a random batch L_t with sampling probability L/N in Poisson subsampling

Compute gradient

For each $i \in L_t$, compute $\mathbf{g}_t(x_i) \leftarrow \nabla_{\theta_t}\mathcal{L}(\theta_t, x_i)$

Clip gradient

$\bar{\mathbf{g}}_t(x_i) \leftarrow \mathbf{g}_t(x_i)/\max\left(1, \frac{\|\mathbf{g}_t(x_i)\|_2}{C}\right)$

Add noise

$\tilde{\mathbf{g}}_t \leftarrow \frac{1}{L}\left(\sum_i \bar{\mathbf{g}}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I})\right)$

Descent

$\theta_{t+1} \leftarrow \theta_t - \eta\tilde{\mathbf{g}}_t$

end for

Output θ_T and compute the overall privacy cost (ε, δ) using a privacy accounting method.

where η is the learning rate.

After the training process completes, the performance of the trained model is evaluated on a held-out test dataset. Since the parameter learning is based on training data, the prediction performance of the trained model is usually good on training data. The ability to also perform well on unseen, test data is referred to as generalization. The performance drop from training to test data is referred to as the generalization gap.

2.3 Differentially Private Stochastic Gradient Descent

Differentially private SGD (DP-SGD) was originally proposed in [1], and still remains the only general backbone for differentially private deep learning that requires no access to additional public data [1, 57, 58, 66]. The pseudocode of DP-SGD is shown in Algorithm 1.

DP-SGD utilizes the Gaussian mechanism to provide DP guarantees. Since the Gaussian mechanism requires an upper-bound on the sensitivity of the data-dependent function, in DP-SGD, the ℓ_2-norm of per-sample gradients $\mathbf{g}_t(x_i) \leftarrow \nabla_{\theta_t}\mathcal{L}(\theta_t, x_i)$ are firstly clipped:

$$\bar{\mathbf{g}}_t(x_i) \leftarrow \mathbf{g}_t(x_i)/\max\left(1, \frac{\|\mathbf{g}_t(x_i)\|_2}{C}\right),$$

where C is the clipping threshold. This clipping step bounds the ℓ_2-sensitivity of $\sum_i \bar{\mathbf{g}}_t(x_i)$ by C and hence the Gaussian noise that scales proportionally to C will achieve DP. After the gradient is noised, it will be used to update the parameters, following the similar update rule to the regular SGD.

Applying the Gaussian mechanism to each gradient update in the way above allows one to obtain the privacy guarantee for each iteration of SGD. To calculate the privacy guarantee corresponding to the overall SGD algorithm, one could apply the advanced composition theorem for DP [13].

Theorem 1. ***Advanced Composition****: For all $\varepsilon, \delta, \delta' \geq 0$, the class of (ε, δ)-differentially private mechanisms satisfies $(\varepsilon', k\delta + \delta')$-differential privacy under k-fold adaptive composition for*

$$\varepsilon' = \sqrt{2k\ln(1/\delta')}\varepsilon + k\varepsilon(e^{\varepsilon} - 1).$$

However, it has been shown in [1] that this generic composition theorem does not yield a tight analysis. Recent works [61, 66, 69] have developed techniques that can produce a much tighter estimate of the overall privacy guarantee for SGD under different subsampling methods and noising mechanisms. Particularly, autodp[1] [61, 69] provides an analytical characterization of privacy guarantees for composite differentially private mechanisms and an efficient implementation to track the guarantees. It is compatible with Poisson subsampling—a subsampling method that is typically assumed in DP-SGD. Hence, in this work, we will use autodp as a default way to calculate the privacy parameters for SGD. Unlike the original paper of DP-SGD [1] that uses fixed-size batches as an approximation, we follow Poisson subsampling strictly in our evaluation for rigorous privacy guarantees.

2.4 Private Aggregation of Teacher Ensembles

PATE is a popular framework for differentially private deep learning when there exists relevant public data. PATE [41, 42] consists of three key ingredients: teacher models, an aggregation mechanism, and a student model.

Teacher models are trained independently on *disjoint* subsets of private data. The ensemble of teacher models is then used to label the public data. An aggregation mechanism is used to aggregate the class predictions produced by each teacher model. Proper noise is injected into the aggregate predictions to ensure that

the labeled public data can achieve certain differential privacy guarantees.

Finally, a student model is trained, usually in a semi-supervised fashion, on public data that is at least partially labeled by an aggregation mechanism. This student model is differentially private as it observes only public data and differentially private labels.

Among the several aggregation mechanisms proposed in prior work [41, 42], the most advanced one is Confident-GNMax Aggregator. It first tests with Gaussian noise of scale σ_1 whether the plurality vote of teacher predictions passes a threshold T. If the test is passed, it returns the class with the most votes after the Gaussian noising of scale σ_2. In this way, priority will be given to labeling public data with sufficiently strong consensus among teacher models.

3 Proposed Approach

3.1 Motivating Example

Let us first see a simple example to motivate the need for smooth learning loss in differentially private learning. Consider the objective function illustrated in Figure 2a, which is the mixture of two centrosymmetric functions $\mathcal{F}_u$ and $\mathcal{F}_v$:

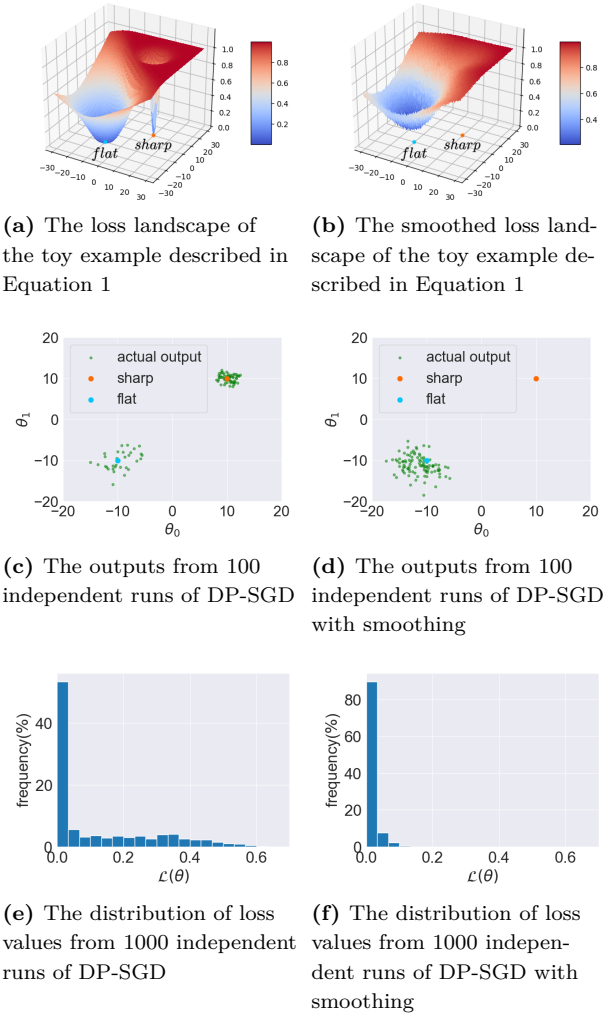
$$\min_{\theta\in\mathcal{R}^2} \mathcal{L}(\theta) = \mathcal{F}_u(\theta)\cdot\frac{\mathcal{E}(\theta,u)}{\mathcal{E}(\theta,u)+\mathcal{E}(\theta,v)} + \mathcal{F}_v(\theta)\cdot\frac{\mathcal{E}(\theta,v)}{\mathcal{E}(\theta,u)+\mathcal{E}(\theta,v)} \tag{1}$$

where $u, v \in \mathcal{R}^2$ are two fixed centers of symmetry, $\mathcal{F}_u(\theta) = \mathcal{S}\left(\frac{\|\theta-u\|_2}{5} - \frac{5}{\|\theta-u\|_2}\right)$, $\mathcal{F}_v(\theta) = \mathcal{S}\left(\frac{2\|\theta-v\|_2}{5} - \frac{5}{2\|\theta-v\|_2}\right)$, the sigmoid function $\mathcal{S}(x) = \frac{1}{1+e^{-x}}$ and $\mathcal{E}(\theta,x) = e^{-\frac{\|\theta-x\|}{2}}$.

As shown in Figure 2a, the loss landscape consists of two local minima with $\mathcal{L}(\theta) \approx 0$: a flat one and a sharp one.

Due to the noise injected to protect privacy, the dynamics of DP-SGD will exhibit large variability. In Figure 2c, we visualize the distribution of θ obtained from 100 independent runs of DP-SGD and we can see that DP-SGD could converge to the neighborhood of both local minima.

The noisy nature of DP-SGD degrades not only the performance but the stability of performance, especially when DP-SGD converges to the sharp local minimum. In Figure 2e, we present the distribution of loss values from 1000 independent runs of DP-SGD. While loss values are near-zero around both local minima, the loss distribution of actual output models has a long tail. The long tail is mainly caused by the the sharp minimum: a slight change near the sharp minimum could lead to a steep increase of loss. In contrast, a change near the flat minimum would result in a small change of loss.



(a) The loss landscape of the toy example described in Equation 1

(b) The smoothed loss landscape of the toy example described in Equation 1

(c) The outputs from 100 independent runs of DP-SGD

(d) The outputs from 100 independent runs of DP-SGD with smoothing

(e) The distribution of loss values from 1000 independent runs of DP-SGD

(f) The distribution of loss values from 1000 independent runs of DP-SGD with smoothing

Fig. 2. A toy example illustrating the motivation for DPLIS

On the other hand, if we can smooth out the sharp local minima, both the performance and the stability of performance will be greatly improved. Figure 2b illustrates the objective after smoothing. We will dive into the details of the smoothing technique later. With enough smoothing, the sharp local minimum will vanish and only the flat one retains. Running DP-SGD with such a smoothed loss will always converge to the neighborhood of the flatter minimum, which enjoys better performances. Moreover, the variance of loss across different runs of DP-SGD can be mitigated.

3.2 Main Algorithm

We leverage the randomized smoothing technique to smooth the learning loss function. Specifically, let $\mathcal{L}(\theta)$ denote the original loss function. For instance, for classification tasks, a standard choice of $\mathcal{L}(\theta)$ is cross-entropy. In general, a randomized smoothing technique convolves the density μ of a random variable with the original loss function:

$$\mathcal{L}_{\text{smooth}}(\theta) = \int \mathcal{L}(\theta')\mu(\theta - \theta')d\theta' = \mathbb{E}_{\Delta\sim\mu}[\mathcal{L}(\theta + \Delta)] \tag{2}$$

The intuition underlying such approaches is that convolving two functions yields a new one that is at least as smooth as the smoothest of the two original functions.

There are various smoothing distributions μ, including Gaussian and uniform distributions on norm balls. We use the Gaussian density function as our default choice of smoothing distribution:

$$\mathcal{L}_{\text{smooth}}(\theta) = \mathbb{E}_{\Delta\sim\mathcal{N}(0,\sigma^2_{\text{smooth}}\mathbf{I})}[\mathcal{L}(\theta + \Delta)] \tag{3}$$

$$= \mathbb{E}_{\theta'\sim\mathcal{N}(\theta,\sigma^2_{\text{smooth}}\mathbf{I})}[\mathcal{L}(\theta')] \tag{4}$$

where σ_{smooth} controls the strength of smoothing: a larger value of σ_{smooth} indicates a smoother loss function. The choice of Gaussian density function is because we found it is more compatible with DP-SGD which adds Gaussian noise into each gradient update step and as we will see later, its combination with DP-SGD leads to a rigorous generalization bound for the trained model.

As a simple illustration of randomized smoothing, in Figure 2b, we present the loss landscape of the toy example from Equation 1 after smoothing. The sharp local minimum vanishes and thus model parameters are effectively guided into the flat local minimum as presented in Figure 2d, which results in better and more stable loss values as shown in Figure 2f. We will formally show that the Gaussian density-based randomized smoothing indeed leads to a smoother loss function in the next section.

In addition to smoothing the loss landscape, there is an alternative way of interpreting our proposed loss function $\mathcal{L}_{\text{smooth}}(\theta)$. Recall that because of the random noise injection, the output model of DP-SGD is intrinsically random. Essentially, the original learning objective, $\min_\theta \mathcal{L}(\theta)$, only focuses on optimizing the performance of a single instantiating of the random output model and ignore the fact that the large randomness of the learning process might severely degrade the model performance. Instead, a more reasonable learning objective for DP-SGD is to minimize the expected performance of the output model. Denoting the distribution of model parameters by $\mathcal{H}(\theta)$, we can write out this new learning objective as:

$$\mathcal{L}_{\text{avg}}(\theta) = \mathbb{E}_{\theta'\sim\mathcal{H}(\theta)}\mathcal{L}(\theta') \tag{5}$$

However, it could be very difficult to analytically characterize $\mathcal{H}(\theta)$, due to the complexity of the model and the learning algorithm. By comparing $\mathcal{L}_{\text{avg}}(\theta)$ with $\mathcal{L}_{\text{smooth}}(\theta)$, we can see that the gist of our proposed smooth loss is to approximate the output model distribution $\mathcal{H}(\theta)$ via a simple Gaussian distribution. Although this could be a crude approximation, it can lead to easily implementable algorithm and at the same time be mindful of the randomness present in DP-SGD.

Note that the second interpretation of our proposed loss can potentially open up new, improved ways of designing the learning objective for DP-SGD. For instance, one can construct a more accurate approximation to $\mathcal{H}(\theta)$ by leveraging the structure of the learning algorithm. However, in this paper, we will just focus on the smoothness interpretation and present an in-depth investigation of the effect of smoothness on the differentially private learning performance.

When it comes to the implementation of the proposed learning loss, there are a few questions to be addressed: (1) how can we calculate the expectation? and (2) how can we choose the smoothing strength σ_{smooth}? We address the first question by using sample averaging to approximate the expectation. As for setting the smoothing strength, the second interpretation of our smoothed loss can provide us with guidelines. In the second interpretation, σ_{smooth} captures the spread of the distribution of the output model parameters. Hence, intuitively, σ_{smooth} should depend on the amount of noise injected by DP-SGD, which further depends on the learning rate, the batch size, and the clipping threshold. Hence, we will factorize σ_{smooth} into the learning rate η, the noise multiplier used in DP-SGD σ, the clipping threshold C, as well as a tuning parameter R, which we call *smoothing radius*. The smoothing radius captures some complex factors that can impact the spread of parameter distribution, like the training data distribution.

Formally, the smoothed loss used in the implementation of DPLIS is given as follows:

$$\min_\theta \frac{1}{K}\sum_{j=1}^{K}\mathcal{L}(\theta + R\cdot\frac{\eta}{L}\cdot\nu_j) \quad \nu_j \sim \mathcal{N}(0, \sigma^2C^2\mathbf{I}) \tag{6}$$

Algorithm 2 DP-SGD with DPLIS

Require: examples $\{x_1, \ldots, x_N\}$, loss function $\mathcal{L}(\theta) = \frac{1}{N}\sum_i \mathcal{L}(\theta, x_i)$, learning rate η, noise multiplier σ, batch size L, clipping threshold C, smoothing radius R, number of smoothing samples K.

Initialize θ_0 randomly

for $t \in [T]$ **do**

Take a random batch L_t with sampling probability L/N in Poisson subsampling

Compute gradient

For each $i \in L_t$, compute

$\mathbf{g}_t(x_i) \leftarrow \frac{1}{K}\sum_{j=1}^{K} \nabla_{\theta_t}\mathcal{L}(\theta_t + R \cdot \frac{\eta}{L} \cdot \mathcal{N}(0, \sigma^2 C^2 \mathbf{I}), x_i)$

Clip gradient

$\bar{\mathbf{g}}_t(x_i) \leftarrow \mathbf{g}_t(x_i) / \max\left(1, \frac{\|\mathbf{g}_t(x_i)\|_2}{C}\right)$

Add noise

$\tilde{\mathbf{g}}_t \leftarrow \frac{1}{L}\left(\sum_i \bar{\mathbf{g}}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I})\right)$

Descent

$\theta_{t+1} \leftarrow \theta_t - \eta \tilde{\mathbf{g}}_t$

end for

Output θ_T and compute the overall privacy cost (ε, δ) using a privacy accounting method.

K is the number of samples that we draw from the smoothing distribution. With a larger K, the loss above will be a better approximation to $\mathcal{L}_{\text{smooth}}(\theta)$ and thus better learning performance could be expected. However, a larger K would at the same makes the learning algorithm more computationally expensive. In general, one can set K to be as large as the computational resource permits. We will present a more detailed study of the effect of K and the smoothing radius in Section 5.

The complete algorithm of DP-SGD with our smoothed loss is provided in Algorithm 2, where the red texts highlight the difference from the original DP-SGD algorithm.

4 Utility Analysis

In this section, we provide the utility analysis for DP-SGD with smoothed loss. We will start by formalizing the smoothness of the loss landscape and further characterize the change in the loss landscape smoothness with randomized smoothing. Moreover, we study the implications of smoothing in improving the stability and generalization of differentially private models. Proofs of lemmas and theorems in this paper can be found in the appendix.

4.1 Smoothness

A commonly used notion for smoothness is based on the Lipschitz constant of the gradient of a function.

Definition 4 (Smoothness)**.** *We say that a function f is β-smooth if*

$$\|\nabla f(x) - \nabla f(y)\| \leq \beta \|x - y\|. \tag{7}$$

β is often referred to as the smoothness constant and a smaller β indicates a smoother function.

The next lemma characterizes the smoothness constant of a function smoothed with randomized smoothing based on Gaussian density.

Lemma 1. *Assume that $\mathcal{L}$ is L-Lipschitz with respect to the ℓ_2 norm. Then, $\mathcal{L}_{smooth}$ in Equation (3) is expected to be L/σ_{smooth}-smooth. In addition, if $\mathcal{L}$ is β-smooth, then $\mathcal{L}_{smooth}$ is expected to be $\min\{L/\sigma_{smooth}, \beta\}$-smooth.*

Lemma 1 indicates that larger Gaussian noise used in randomized smoothing will lead to a smoother loss landscape. Moreover, note that the first part of Lemma 1 makes no assumption about the smoothness of the original loss function. Hence, even when the original function is unsmooth (i.e., unbounded smooth constant), randomized smoothing can still lead to a smooth function. Indeed, unsmooth loss is quite common in deep learning. For instance, when the ReLU activation function is used, the resulting deep net has unbounded smoothness constant. The second part of the theorem implies that the smoothness constant of $\mathcal{L}_{\text{smooth}}$ is always less than that of $\mathcal{L}$. In other words, if the original loss function is already smooth, randomized smoothing can only make it smoother.

4.2 Stability

With the characterization of smoothness, we can reason about the effect of smoothness on the stability of differentially private learning, i.e., the consistency of model performance across different runs. Particularly, we will use the sum of the expected squared l_2-norm of the gradient across all iterations to help measure the stability. Gradients induce the changes in the model parameters and if each gradient has a large variance, then the model parameters will also have a large variance. Hence, the sum of the expected squared l_2-norm of the gradient reflects the total variations exhibited during training.

Theorem 2 provides a bound on the total expected squared l_2-norm of the gradients in terms of the smoothness constant with a simplification through ignoring the effect of gradient clipping. The proof follows from the convergence proof of SGD in [19, 35].

Theorem 2. *Given an arbitrary β-smooth learning loss $\mathcal{L}$ and a learning rate η, assuming $\mathbb{E}[\nabla\mathcal{L}(\theta, x_i)] = \mathbb{E}[\nabla\mathcal{L}(\theta)]$ and $\mathbb{E}[\|\nabla\mathcal{L}(\theta, x_i) - \nabla\mathcal{L}(\theta)\|^2] \leq \sigma^2$ for some parameter $\sigma \leq 0$, we have*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\|\nabla\mathcal{L}(\theta_t)\|^2] \leq \frac{2\mathbb{E}[\mathcal{L}(\theta_1)] - 2\mathbb{E}[\mathcal{L}(\theta^*)]}{\eta(2-\eta\beta)T} + \frac{\eta\beta\sigma^2}{(2-\eta\beta)}$$

If we use a monotonically decreasing learning rate η and $\eta = \omega(\frac{1}{T})$, then when $T \to \infty$, the second term on the right-hand side will dominate. Since randomized smoothing can effectively reduce the smoothness constant, the left-hand side for the smoothed loss will be less than that for the original loss, which further implies that DP-SGD with DPLIS is more stable than the vanilla DP-SGD.

4.3 Generalization

We now switch up to another performance metric for differentially private learning—generalizability of the trained model.

We show with the following theorem that randomized smoothing may help to close generalization gaps. While the same proof idea can be generalized to Poisson subsampling, a simplification is made in the following theorem by considering DP-SGD that uses fixed-size batches instead. This is to derive a more succinct bound that offers more insights.

Theorem 3 (Generalization by Smoothness)**.** *Let $\hat{\mathcal{L}}(\theta)$ be the expected loss on the actual distribution, $\mathcal{L}(\theta)$ be the corresponding average loss computed on the training data and $\mathcal{L}(\theta, x)$ be the corresponding loss on sample x. Let $\mathcal{L}_{train}(\theta, x)$ be the training loss used in DP-SGD on sample x, which may differ from $\mathcal{L}(\theta, x)$. Assume $\mathcal{L}(\theta, x)$ is L-Lipschitz for every x and $\mathcal{L}_{train}(\theta, x)$ is β-smooth every x, then we have*

$$|\mathbb{E}[\hat{\mathcal{L}}(\theta) - \mathcal{L}(\theta)]| \leq \frac{2\eta CLT}{n} \cdot (1+\eta\beta)^{T-1}, \qquad (8)$$

where θ is the final parameter from DP-SGD with learning rate η, clipping threshold C, noise multiplier σ, total number of steps T and total number of training data n. The expectations are taken over both the randomness of DP-SGD and the draw of training data.

Theorem 3 indicates that the generalization error of a model (i.e., left-hand side of Eqn. 8) depends on the smoothness of the loss function used for training the model (i.e., β in the right-hand side of Eqn. 8) and a smoother training loss can potentially lead to a smaller generalization error. The theorem justifies the advantage of using randomized smoothing during training.

The proof is largely inspired by [20] and is left to the appendix. The key idea of the proof is to leverage a classic result in learning theory that the generalization gap $\mathbb{E}[\hat{\mathcal{L}}(\theta) - \mathcal{L}(\theta)]$ can be bounded by s when the learning algorithm is s-uniformly stable (i.e. $\mathbb{E}[\mathcal{L}(\theta, x)]$ may differ at most s when one sample in the training set is replaced). Let D and D' be respectively training set before and after the replacement. Since $\mathcal{L}(\theta, x)$ is L-Lipschitz, the difference $\mathbb{E}[\mathcal{L}(\theta_D, x)] - \mathbb{E}[\mathcal{L}(\theta_{D'}, x)]$ is bounded by $L \cdot \mathbb{E}[\|\theta_D - \theta'_D\|]$, where θ_D and $\theta_{D'}$ represent the models trained on D and D', respectively. Thus, we can analyze $\mathbb{E}[\|\theta_D - \theta'_D\|]$ instead. Fixing the randomness of initial parameters, Gaussian noise, and batch selection, we can analyze how θ_D and θ'_D depart from each other through DP-SGD updates using the smoothness of $\mathcal{L}_{\text{train}}$. Intuitively, the smoother the training objective is, the less sensitive individual iterations of DP-SGD would be to changes of training points. At last, we take expectation over the randomness above to obtain the final bound of uniform stability, which is then transferred into the generalization bound.

5 Evaluation

We would like to answer the following five questions with empirical evaluation:

1. How does DPLIS perform compared to vanilla DP-SGD?
2. Is the performance sensitive to the newly introduced hyper-parameters (i.e., K and smoothing radius)?
3. Does randomized smoothing indeed lead to smoother loss landscapes for DNNs?
4. Does randomized smoothing improve the stability?
5. Will the gains from smoothing remain combining other improvement strategies?

In addition, we present how the smoothing idea can be extended to improve PATE—a popular framework for differentially private deep learning with public data.

5.1 Experimental Setup

Datasets and Models. To answer question (1), we evaluate DPLIS on various learning tasks. We first demonstrate DPLIS's performance improvement on two classic vision benchmarks: **MNIST** and **CIFAR-10**. On these two benchmarks, models are trained entirely from scratch, which we refer to as *non-transfer settings*. MNIST [29] is one of the most commonly used benchmark datasets in deep learning containing 70000 handwritten digit images. CIFAR-10 [28] is another classic benchmark for image classification. It consists of 60000 images from 10 different classes with 6000 images each. We evaluate two different architectures on MNIST and CIFAR-10: multilayer perceptron (MLP) and convolution neural network (CNN). The MLP is comprised of two hidden layers with 512 (1536) neurons and 128 (128) neurons for MNIST (CIFAR-10) all using ReLU activation. The architecture of the CNN is inherited from the official tutorial of tensorflow/privacy[2].

Besides non-transfer settings, given the access to pre-trained feature extractors, we also evaluate DPLIS on more challenging datasets, which we refer to as *transfer settings*. Specifically, we evaluate DPLIS in three transfer settings.

- **ImageNet→CIFAR-100**: ImageNet [47] denotes the image classification benchmark in ImageNet Large Scale Visual Recognition Challenge (ILSVRC). It includes a training set with over 1.2 million images and a validation set of 50000 images, all in full resolution, spanning 1000 different classes. CIFAR-100 [28] is similar to CIFAR-10 but with 100 different classes. In this setting, we use a ImageNet pre-trained EfficientNet-b0[3] [54] to extract features. After average-pooling, the extractor provides 1280-dimensional feature vectors, on which an MLP with two 256-neuron hidden layers and ReLU activation is trained to classify CIFAR-100.
- **CelebA→PubFig83**: PubFig83[44] is a dataset of 13838 facial images from 83 public figures with at least 100 images per identity. In our evaluation, we use a pre-processed version of PubFig83[4] [9], where images are aligned by the position of the eyes. 50 images from every identity are taken into the testing set with a total of 4150 images, and the remaining 9688 images constitute the training set. CelebA[36] is a large-scale dataset of 202599 facial images corresponding to 10177 identities. We remove all 44 identities with only one corresponding image from CelebA, which results in a total of 202555 images and 10133 identities. In this setting, the pre-training is accomplished on this dataset using 10133-way face identification with one image from every identity preserved for validation. We choose ResNet-50[21] as the backbone of pre-trained feature extractors, following the implementation of face.evoLVe[5]. After pre-training, average-pooling is applied to the outputs of the last convolution layer to form features with 2048 dimensions, and an MLP with one 32-neuron hidden layer and ReLU activation is trained on such features.
- **CelebA$_-$→CelebA$_{1000}$**: In CelebA$_-$→CelebA$_{1000}$ setting, 1000 identities with exactly 30 corresponding facial images are picked to form the private dataset, namely CelebA$_{1000}$, while the remaining 9133 identities constitutes the dataset for pre-training, namely CelebA$_-$. The pre-training on CelebA$_-$ is similar to the one in CelebA→PubFig83 setting, with one image from every identity preserved for validation. For the private dataset CelebA$_{1000}$, 25 images from every identity are taken by the training set, and the other 5 images are taken by the testing set, which leads to a training set of size 25000 and a testing set of size 5000. The architecture is the same as CelebA→PubFig83 setting.

Beyond vision tasks, we also evaluate DPLIS on an NLP task, namely next word prediction on **Reddit Comments** (Reddit). Reddit Comments [2] is a collection of Reddit posts. Following the setup of [25], we randomly sample 10000 comments as training data and 5000 comments as testing data, and pre-train the model on an additional public data source, Brown corpus [15], without differential privacy for warm-starting. Following [38], we choose to use a pre-selected dictionary from public data instead of to use heavy hitter estimation[45] to generate one from private data. Our dictionary is composed of a total of 36743 words, containing all words with a frequency of at least two from Brown corpus.

We evaluate both MLP and LSTM models for the task. The MLP takes the last 20 tokens as its input and contains three hidden layers with respectively 500, 250, and 50 neurons. The LSTM model takes a series of words as input and embeds them individually into a learned 256-dimensional space. The embedded characters are then processed through an LSTM module with 96 nodes. Finally, the output of the LSTM module is sent to a fully-connected layer for word predictions.

Training Setups. For all the reported results, all hyper-parameters for DP-SGD are selected to achieve the best performance of vanilla DP-SGD. DPLIS then sets the exactly same value for the part of hyperpa-

Table 1. Hyper-parameters

setting	L	$\frac{\eta}{L}$	η	σ	C
MNIST	**256**	$\mathbf{6 \times 10^{-4}}$	**0.1536**	**1.1**	**1.0**
CIFAR-10	**256**	$\mathbf{6 \times 10^{-4}}$	**0.1536**	**1.1**	**1.0**
ImageNet→CIFAR-100	**256**	$\mathbf{4 \times 10^{-4}}$	**0.1024**	**3.0**	**1.0**
CelebA→PubFig83	**256**	$\mathbf{6 \times 10^{-3}}$	**1.536**	**2.0**	**1.0**
CelebA$_-$→CelebA$_{1000}$	**256**	$\mathbf{6 \times 10^{-3}}$	**1.536**	**1.3**	**1.0**
Reddit	**64**	$\mathbf{10^{-3}}$	**0.064**	**1.1**	**1.0**

rameters that appear in DP-SGD. The exact choice of hyper-parameters are presented in Table 1, where L is (expected) batch size, η is learning rate, σ is noise multiplier and C is clipping threshold. Batches are sampled through Poisson sampling with probability $\frac{L}{N}$, where N is the size of the training set.

For DPLIS, unless otherwise specified, we set $K = 10$ in both non-transfer and transfer settings and $K = 20$ for next word prediction on Reddit. We present results for radius $R = 10$ and $R = R_{\text{best}}$, where R_{best} denotes the radius that achieves the best performance under corresponding privacy budgets.

In presenting privacy budgets, we fix δ in $(\varepsilon, \delta)-$differential privacy to be 10^{-5} and present only the corresponding ε. Each row in Table 2, Table 3 and Table 5 corresponds to results from a single run.

5.2 Performance Evaluation

Results in Non-transfer Settings. As shown in Table 2, using DPLIS improves test accuracy of DP-SGD by $0.46\% \sim 5.88\%$ with $R = R_{\text{best}}$ and $0.06\% \sim 3.88\%$ with $R = 10$ in non-transfer settings.

An interesting phenomenon is that the test accuracy of vanilla DP-SGD occasionally drops drastically when training proceeds. As a concrete example, when we train MLP on CIFAR-10, the test accuracy starts to degrade as early as before $\varepsilon = 5.01$. Though it seems like overfitting, it is actually not since no increasing generalization gap is observed. We will further discuss this in Section 5.4. This issue, if not fully addressed, is greatly alleviated by randomized smoothing. When training proceeds, no significant drop on test accuracy is observed for DPLIS.

Results in Transfer Settings. As in Table 3, while pre-trained features already lift the utility of vanilla DP-SGD, using DPLIS can further improve test accuracy of DP-SGD by $1.23\% \sim 5.81\%$ with $R = R_{\text{best}}$ and $0.56\% \sim 5.81\%$ with $R = 10$.

Firstly, pre-trained features greatly boost capabilities of vanilla DP-SGD in a sense that now it can perform fairly well within meaningful privacy budgets on much more difficult tasks. Table 4 contains a summary of training set statistics, where N_{train} denotes the size of training set, M denotes the size of label set and $\frac{N_{train}}{M}$ denotes the average number of samples per label. CIFAR-100, PubFig83 and CelebA$_{1000}$ are considered much harder than MNIST and CIFAR-10, as they have more label classes, fewer samples per class, and richer details in images. However, with a pre-trained EfficientNet-b0 as the feature extractor, an accuracy of 46.78% is achieved on CIFAR-100 by DP-SGD without smoothing in a very tight privacy budget $\varepsilon = 0.86$, which is already comparable to 49.70% on CIFAR-10 by training a CNN from scratch with $\varepsilon = 1.99$. Clearly, it is beneficial to have pre-trained features when using DP-SGD.

Secondly, even with strong pre-trained features, randomized smoothing remains a substantial improvement to model performance when applied in DP-SGD. This observation is consistent in all three settings, regardless of whether the feature extractor is pre-trained on a dataset with a broader scope (ImageNet→CIFAR-100), a dataset with domain shift (CelebA→PubFig83), or even a dataset with a similar distribution (CelebA$_-$→CelebA$_{1000}$). With the presence of randomized smoothing, test accuracy is raised by at least 3% in many cases and around 5% in a few. The effectiveness of randomized smoothing in improving utility does not seem to be weakened by access to pre-trained features.

Results on Next Word Prediction. In Table 5, applying DPLIS results in a significant performance improvement, which further supports the effectiveness of our approach. Perplexity is reduced by at least 50 in all cases for both $R = 10$ and $R = R_{\text{best}}$.

Similar to the image-domain tasks in non-transfer settings, we observe degrading performance for the LSTM model when using DP-SGD without smoothing. The perplexity is indeed increasing gradually when training proceeds. While we will further discuss the potential cause of this issue in Section 5.4, we can see here that it is again addressed by randomized smoothing successfully. Perplexity decreases and utility improves through training with the presence of randomized smoothing.

Table 2. Evaluation in Non-transfer Settings

dataset	model	smoothing	test accuracy			
			$\varepsilon = 1.99$	$\varepsilon = 5.01$	$\varepsilon = 7.01$	$\varepsilon = 10.00$
MNIST	MLP	No	92.60%	93.92%	94.11%	93.35%
		$R = 10$	92.66%(+0.06%)	94.78%(+0.86%)	94.84%(+0.73%)	95.19%(+1.84%)
		$R = R_{\text{best}}$	93.58%(+0.98%)	95.94%(+2.02%)	96.17%(+2.06%)	96.00%(+2.65%)
	CNN	No	95.08%	95.92%	96.07%	95.89%
		$R = 10$	96.62%(+1.54%)	97.49%(+1.57%)	97.52%(+1.45%)	98.56%(+2.67%)
		$R = R_{\text{best}}$	97.31%(+2.23%)	98.01%(+2.09%)	98.48%(+2.41%)	98.90%(+3.01%)
CIFAR-10	MLP	No	43.88%	43.16%	41.95%	41.07%
		$R = 10$	44.09%(+0.21%)	44.48%(+1.32%)	44.73%(+2.78%)	44.95%(+3.88%)
		$R = R_{\text{best}}$	44.34%(+0.46%)	46.90%(+3.74%)	46.92%(+4.97%)	46.95%(+5.88%)
	CNN	No	49.70%	58.48%	60.46%	61.37%
		$R = 10$	50.09%(+0.39%)	60.39%(+1.91%)	62.09%(+1.63%)	63.22%(+1.85%)
		$R = R_{\text{best}}$	50.85%(+1.15%)	61.75%(+3.27%)	62.32%(+1.86%)	64.73%(+3.36%)

Table 3. Evaluation in Transfer Settings

setting	smoothing	test accuracy				
ImageNet→ CIFAR-100		$\varepsilon = 0.39$	$\varepsilon = 0.54$	$\varepsilon = 0.66$	$\varepsilon = 0.77$	$\varepsilon = 0.86$
	No	34.75%	42.34%	44.52%	46.00%	46.78%
	$R = 10$	35.73%(+0.98%)	43.32%(+0.98%)	46.59%(+2.07%)	49.02%(+3.02%)	49.70%(+2.92%)
	$R = R_{\text{best}}$	35.98%(+1.23%)	43.76%(+1.42%)	48.00%(+3.48%)	51.04%(+5.04%)	52.04%(+5.26%)
CelebA→ PubFig83		$\varepsilon = 2.84$	$\varepsilon = 4.06$	$\varepsilon = 5.04$	$\varepsilon = 5.89$	$\varepsilon = 6.64$
	No	41.20%	53.45%	55.95%	59.47%	62.17%
	$R = 10$	41.76%(+0.56%)	54.53%(+1.08%)	61.76%(+5.81%)	62.84%(+3.37%)	64.84%(+2.67%)
	$R = R_{\text{best}}$	43.16%(+1.96%)	57.16%(+3.71%)	61.76%(+5.81%)	64.27%(+4.80%)	66.24%(+4.07%)
CelebA$_{-}$→ CelebA$_{1000}$		$\varepsilon = 4.88$	$\varepsilon = 6.06$	$\varepsilon = 7.12$	$\varepsilon = 8.03$	$\varepsilon = 8.87$
	No	19.68%	25.74%	31.54%	34.12%	36.44%
	$R = 10$	21.46%(+1.78%)	29.86%(+4.12%)	34.32%(+2.78%)	38.18%(+4.06%)	40.32%(+3.88%)
	$R = R_{\text{best}}$	22.54%(+2.86%)	30.38%(+4.64%)	35.58%(+4.04%)	38.88%(+4.76%)	41.08%(+4.64%)

Table 4. Summary of Training Set Statistics

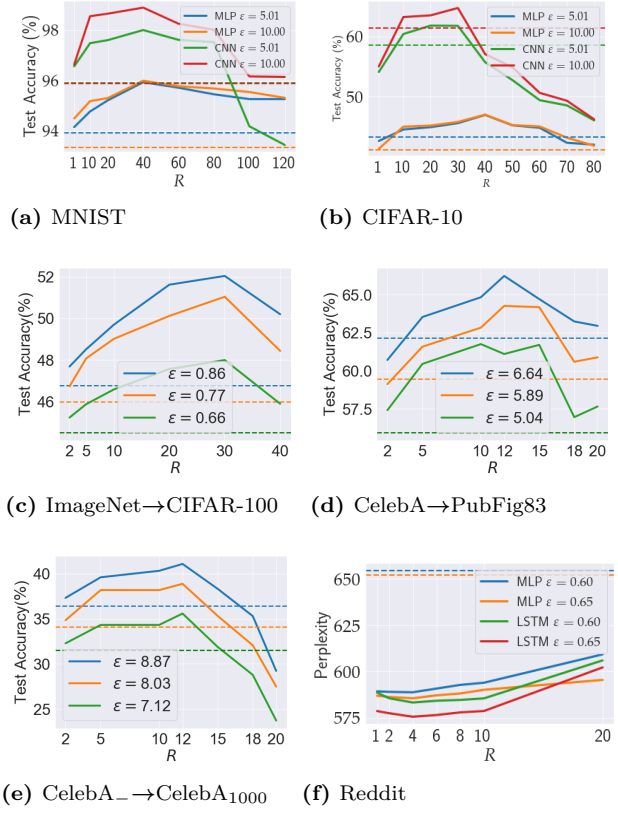
dataset	N_{train}	M	$\frac{N_{train}}{M}$
MNIST	60000	10	6000
CIFAR-10	50000	10	5000
CIFAR-100	50000	100	500
PubFig83	9688	83	≈ 117
CelebA$_{1000}$	25000	1000	25

5.3 Hyper-parameters for Randomized Smoothing

Applying randomized smoothing introduces only two scalar hyper-parameters, smoothing radius R and the number of smoothing samples K. With existing private hyper-parameter selection techniques[34, 51], adding two additional scalar hyper-parameters to existing hyper-parameters in DP-SGD is in fact not too much of a burden on privacy budgets, even if they are treated naively through grid search. Nevertheless, we will show in this section that by digging into the effects of R and K to model performance, their selection can be done properly with less or even no additional privacy budget.

Table 5. Evaluation of Next Word Prediction on Reddit

dataset	model	smoothing	perplexity		
			$\varepsilon = 0.55$	$\varepsilon = 0.60$	$\varepsilon = 0.65$
Reddit	MLP	No	692.85	654.66	652.30
		$R = 10$	638.10(−54.75)	593.92(−60.74)	590.11(−62.19)
		$R = R_{\text{best}}$	622.81(−70.04)	588.76(−65.90)	585.50(−66.80)
	LSTM	No	1157.42	2909.73	2981.27
		$R = 10$	688.95(−468.47)	585.47(−2324.26)	578.63(−2402.64)
		$R = R_{\text{best}}$	683.19(−474.23)	583.23(−2326.50)	575.54(−2405.73)



(a) MNIST (b) CIFAR-10

(c) ImageNet→CIFAR-100 (d) CelebA→PubFig83

(e) CelebA$_{-}$→CelebA$_{1000}$ (f) Reddit

Fig. 3. Test accuracy v.s. smoothing radius R

Table 6. R_{best} in Non-transfer Settings

dataset	model	R_{best}			
		$\varepsilon = 1.99$	$\varepsilon = 5.01$	$\varepsilon = 7.01$	$\varepsilon = 10.00$
MNIST	MLP	40	40	40	40
	CNN	40	40	40	40
CIFAR-10	MLP	30	40	40	40
	CNN	30	30	30	30

Table 7. R_{best} in Transfer Settings

setting	R_{best}				
ImageNet→ CIFAR-100	$\varepsilon = 0.39$	$\varepsilon = 0.54$	$\varepsilon = 0.66$	$\varepsilon = 0.77$	$\varepsilon = 0.86$
	5	12	30	30	30
CelebA→ PubFig83	$\varepsilon = 2.84$	$\varepsilon = 4.06$	$\varepsilon = 5.04$	$\varepsilon = 5.89$	$\varepsilon = 6.64$
	15	12	10	12	12
CelebA$_{-}$→ CelebA$_{1000}$	$\varepsilon = 4.88$	$\varepsilon = 6.06$	$\varepsilon = 7.12$	$\varepsilon = 8.03$	$\varepsilon = 8.87$
	12	12	12	12	12

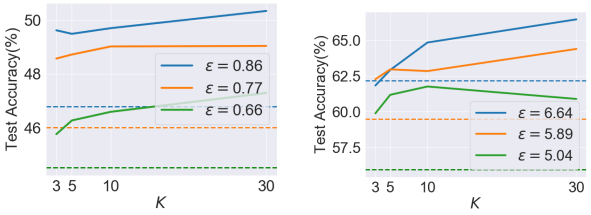
Table 8. R_{best} for Next Word Prediction on Reddit

dataset	model	R_{best}		
		$\varepsilon = 0.55$	$\varepsilon = 0.60$	$\varepsilon = 0.65$
Reddit	MLP	4	4	4
	LSTM	4	4	4

Effect of R. To understand the effect of R, we evaluate randomized smoothing with various R under $K = 20$ for next word prediction on Reddit and under $K = 10$ for other settings. The results are presented in Figure 3. We also report the values of R_{best} in Table 6, Table 7 and Table 8 respectively for different settings. Each color corresponds to a specific setting with a specific value of ε, with the full line in that color denoting performance with smoothing using different R and the dotted line in that color denoting performance without smoothing.

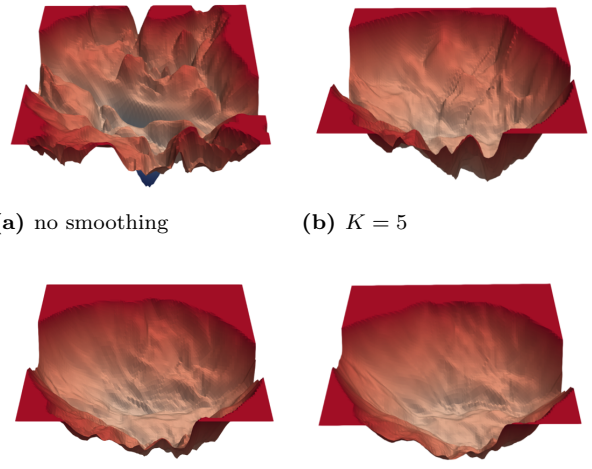
Regarding the effect of R to model performance, we observe a useful property here: model performance is approximately a unimodal function of R, which means it is monotonically increasing for $R < R_{\text{best}}$ and monotonically decreasing for $R > R_{\text{best}}$. With this property, the set of R that yields considerable improvements forms an interval that is always fairly wide in all our results, which is friendly to private hyper-parameter selection.

Furthermore, benefiting from our factorization of σ_{smooth}, even though σ_{smooth} may vary greatly, the interval for R with solid improvements is more or less



(a) ImageNet→CIFAR-100 **(b)** CelebA→PubFig83

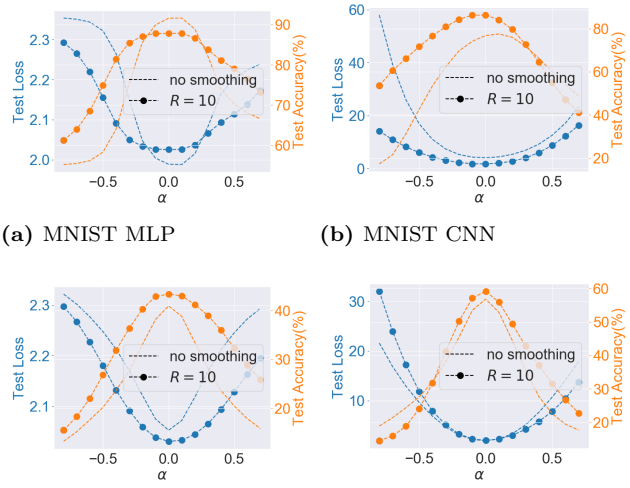
Fig. 4. Test accuracy v.s. number of smoothing samples K



(a) no smoothing **(b)** $K = 5$

(c) $K = 10$ **(d)** $K = 20$

Fig. 5. Loss surfaces on 1% training data of CIFAR-10



(a) MNIST MLP **(b)** MNIST CNN

(c) CIFAR-10 MLP **(d)** CIFAR-10 CNN

Fig. 6. Sharpness visualization

of the same scale across different settings. As a result, while R_{best} is different from one setting to another, smoothing DP-SGD with a single radius setting $R = 10$ suffices to considerably outperform vanilla DP-SGD across all settings as shown in Table 2, Table 3 and Table 5. Thus, as a rule of thumb, simply selecting $R = 10$ is likely a good choice.

Effect of K. To understand the role of the hyper-parameter K, we evaluate randomized smoothing with various K under a fixed radius $R = 10$ in two settings, ImageNet→CIFAR-100 for object classification and CelebA→PubFig83 for face identification. The results are presented in Figure 4. For each setting, we present results corresponding to three different privacy budget ε. For each ε, the test accuracy obtained with different K is plotted as a full line and the test accuracy without smoothing under the same ε is plotted as a horizontal dotted line in the same color.

Overall, model performance benefits from a larger K, which makes the selection of K a simple trade-off between utility and computation time. Such trade-off can be easily addressed without tuning by selecting the largest K with acceptable training time. This phenomenon is fairly intuitive given that a larger K implies a better approximation of $\mathcal{L}_{\text{smooth}}(\theta)$. Figure 5 supports this claim, in which we visualize the effect of K to the loss surface of a ResNet-56 published by [32] along with 3D loss visualization code[6]. Besides, even with K as small as 3, randomized smoothing can already improve the performance of DP-SGD considerably, as shown in Figure 4.

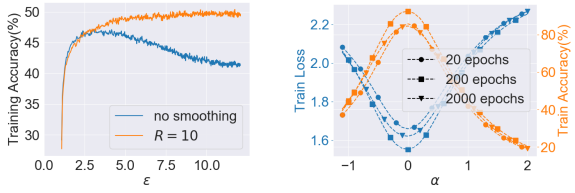
5.4 Justification of Smoothing

When designing DPLIS, one of our motivations is to explicitly incorporate the preference to flatter regions, which is supported by theoretical results in Section 4. In this section, we provide further supports by examining empirically whether DP-SGD is guided towards flatter regions with randomized smoothing.

One way of examination is through visualization, which is intuitive but considers limited dimensions. We adapt filter normalized plots from [26] for visualization, which is designed to remove apparent differences in geometry caused by scaling filter weights. To visualize the sharpness of loss landscape around a given parameter θ, one begins with a random Gaussian direction d with the same dimensions as θ. Then each filter f_d in d will be normalized by a factor of $\frac{\|f_d\|}{\|f_\theta\|}$, so that it will have the same norm as the corresponding filter f_θ in θ. At last, with the filter-normalized vector d, one can plot

Table 9. $(C_\epsilon; A)$-sharpness

dataset	model	smoothing	$(C_\epsilon; A)$-sharpness
MNIST	MLP	No	0.8626
		$R = 10$	0.5454
	CNN	No	139.8159
		$R = 10$	49.6169
CIFAR-10	MLP	No	1.0375
		$R = 10$	0.0000
	CNN	No	563.5351
		$R = 10$	340.5471



(a) Training accuracy v.s. ε **(b)** Sharpness visualization of SGD

Fig. 7. An explanation to the cause of degrading performance issue of DP-SGD

out test accuracy and test loss corresponding to models parameters on the segment $\theta + \alpha \cdot d$ with $\alpha \in [-0.8, 0.8]$.

We visualize models trained with and without randomized smoothing in three different settings. The results are presented in Figure 6. Overall, the visualized region around the parameter obtained by DP-SGD with smoothing tends to be flatter than the one around the parameter obtained by vanilla DP-SGD. Though the visualization is conducted along a single direction, it gives us a rough idea of the effect of DPLIS.

Another way of examination is through numerical sharpness metrics, which is less intuitive but able to capture patterns from more dimensions. Here, we follow the definition of $(C_\epsilon; A)$-sharpness from [26].

Definition 5. ***$(C_\epsilon; A)$-sharpness****: Given $\theta \in \mathcal{R}^n$, $\epsilon > 0$ and $A \in \mathcal{R}^{n \times p}$, the $(C_\epsilon; A)$-sharpness of $\mathcal{L}$ at θ is*

$$\frac{\left(\max_{y \in C_\epsilon(\theta)} \mathcal{L}(\theta + Ay)\right) - \mathcal{L}(\theta)}{1 + \mathcal{L}(\theta)} \times 100,$$

where $C_\epsilon(\theta) = \{z \in \mathcal{R}^p : |z_i| \leq \epsilon \left(\left|\left(A^+\theta\right)_i\right| + 1\right)\}$ and A^+ denotes pseudo-inverse of A.

We set A to be the identity matrix I_n following their default setting and set ϵ to be 0.0045. The results are presented in Table 9, where $(C_\epsilon; A)$-sharpness is always smaller with randomized smoothing, suggesting the success of our design in guiding DP-SGD towards favorable flatter regions.

In Section 5.2, we briefly mention the degrading performance issue of vanilla DP-SGD when it is observed in Table 2 and Table 5, where model performance decreases gradually as training proceeds. Here we propose an explanation for the main cause of such an issue, which separates it from overfitting in machine learning, and we argue that guiding DP-SGD towards flatter regions via randomized smoothing can indeed resolve this issue.

In Figure 7a, we have curves of training accuracy on CIFAR-10 plotted for DP-SGD with and without randomized smoothing. We notice that not only the performance on the test set but also the performance on the training set can be degrading when using vanilla DP-SGD. The degrading training performance, instead of the enlarged generalization gap (i.e., overfitting), is the major cause for degrading test performance.

We further propose a hypothesis: degrading training performance is a joint effect of a growing favor to sharper minima and the noisy nature of DP-SGD. In Figure 7b, we visualize training accuracy and training loss of three checkpoints from a training process with SGD and gradient clipping, where we observe that SGD may start guiding the parameter towards sharper minima from some point of training. Sharper minima are considered less noise-tolerant and degrading training accuracy occurs when starting at some point they fail to tolerate noise introduced by DP-SGD. Randomized smoothing resolves this issue by mitigating the tendency for sharper minima with an explicit preference to flatter regions.

5.5 Stability of Performance

Vanilla DP-SGD is usually considered unstable as variations of performance among independent runs are non-negligible. In this section, we show that randomized smoothing improves not only the performance of DP-SGD, but also its stability.

Table 10 contains our evaluation results. For each cell in the table, range and standard deviation of test accuracy across 5 runs are reported. Both standard deviation and range can be reduced significantly after using randomized smoothing, which indicates that randomized smoothing helps with the stability of performance.

Table 10. Stability of Performance

dataset	model	smoothing	standard deviation		range	
			$\varepsilon = 5.01$	$\varepsilon = 10.00$	$\varepsilon = 5.01$	$\varepsilon = 10.00$
MNIST	MLP	No	0.184%	0.339%	0.55% (93.51%, 94.06%)	1.02% (93.25%, 94.27%)
		$R = 10$	0.088%	0.184%	0.25% (94.72%, 94.97%)	0.56% (95.36%, 95.92%)
		$R = 40$	0.068%	0.088%	0.18% (95.92%, 96.10%)	0.25% (95.90%, 96.15%)
CIFAR-10	CNN	No	0.577%	0.427%	1.61% (57.27%, 58.88%)	1.18% (60.65%, 61.83%)
		$R = 10$	0.444%	0.360%	1.24% (60.18%, 61.42%)	0.92% (62.66%, 63.58%)
		$R = 30$	0.332%	0.196%	0.96% (61.22%, 62.18%)	0.58% (64.25%, 64.83%)

Table 11. Evaluation when combined with Tempered Sigmoid

dataset	model	smoothing	test accuracy			
			$\varepsilon = 1.99$	$\varepsilon = 5.01$	$\varepsilon = 7.01$	$\varepsilon = 10.00$
CIFAR-10	CNN	No	50.04%	57.01%	58.07%	57.46%
		$R = 10$	50.38%(+0.34%)	58.18%(+1.17%)	59.58%(+1.51%)	59.13%(+1.67%)
		$R = R_{\text{best}}$	50.38%(+0.34%)	59.17%(+2.16%)	59.93%(+1.86%)	60.06%(+2.60%)

Table 12. Evaluation of Smoothing for PATE

dataset	parameter	smoothing	test accuracy			
SVHN	$\sigma_2 = 40$		$\varepsilon = 7.44$	$\varepsilon = 11.08$	$\varepsilon = 14.08$	$\varepsilon = 16.75$
		No	82.56%	84.44%	85.79%	86.77%
		Yes	84.06%(+1.50%)	85.64%(+1.20%)	86.96%(+1.17%)	87.39%(+0.62%)
	$\sigma_2 = 80$		$\varepsilon = 3.78$	$\varepsilon = 5.51$	$\varepsilon = 6.90$	$\varepsilon = 8.10$
		No	75.65%	76.77%	81.95%	83.28%
		Yes	77.84%(+2.19%)	82.06%(+5.29%)	83.62%(+1.67%)	84.64%(+1.36%)
	$\sigma_2 = 100$		$\varepsilon = 3.15$	$\varepsilon = 4.57$	$\varepsilon = 5.70$	$\varepsilon = 6.68$
		No	70.51%	75.75%	79.19%	80.50%
		Yes	73.92%(+3.41%)	79.56%(+3.81%)	81.69%(+2.50%)	82.36%(+1.86%)

5.6 Compatibility of DPlis

Since DPLIS tailors the training objectives through smoothing, it is by design complementary to and therefore naturally compatible with most if not all existing strategies. As a proof of concept, we show empirically in this section that the gains from DPLIS remain when combined with Tempered Sigmoid [43], which modifies the activation functions.

The results are included in Table 11. Following [43], we replace ReLU activations with tanh, the default setting of Tempered Sigmoid. Other setups are the same as in Section 5.1. The values of R_{best} are $10, 15, 20, 15$ respectively for $\varepsilon = 1.99, 5.01, 7.01, 10.00$.

On CIFAR-10 using a CNN with Tempered Sigmoid activations, DPLIS improves the test accuracy by $0.34\% \sim 2.60\%$ with $R = R_{\text{best}}$ and $0.34\% \sim 1.67\%$ with $R = 10$, which corroborates the compatibility of DPLIS with other strategies.

5.7 Extension to PATE

In this section, we show that the idea of smoothing learning loss can be extended easily to PATE for improved utility as well. The core of PATE is noisy data labeling. Intuitively, the label noise manifests itself as parameter noise during training, which can be mitigated by smoothing.

Experimental Setup. We evaluate PATE on SVHN[40] benchmark following mostly setups from [41, 42]. Detailed experimental setup is included in Appendix D.

Experimental Results. As shown in Table 12, smoothing improves test accuracy of PATE by 0.62% $\sim$ 5.29%. PATE benefits from smoothing because of the increased tolerance to the noise in privacy-preserving labels. In Table 13, we report the accuracy of privacy-preserving labeling P_{correct}. We notice that in most cases, the improvement from smoothing is greater for settings with lower P_{correct}, which corroborates the ability of smoothing to handle label noise.

6 Related Work

Privacy Attacks. Various privacy attacks motivate the development of DP. Privacy attacks against machine learning models aim to learn private information about the training data. Membership inference and model inversion are two major attacks of interest in the literature. Membership inference attacks aim to determine whether a given individual's data record is included in the training set of the model [8, 33, 37, 49, 50]. On the other hand, model inversion attacks aim to reconstruct the sensitive features of the training data records [16, 17, 48, 65, 68]. Additionally, model inversion attack can be further refined into property inference attack, where the attacker can speculate whether there is a certain statistical property in the training dataset [3, 18, 59]. While DP provides a strong theoretical guarantee against these privacy attacks, it will typically cause unbearable utility degradation for the trained models [46, 60]. Our work can be applied to obtain a better utility-privacy tradeoff against privacy attacks.

Differentially Private Optimization. Differentially private optimization is one of the most important applications of DP and has gone through careful studies during the past decade. For convex optimization, there is a line of works with different trade-offs between utility, privacy, and usability. [7] proposed the classic techniques of output perturbation and objective perturbation. A thorough analysis of the techniques appear in [24] and two variants were proposed in [23, 64]. Several mechanisms [27, 53, 56] were proposed for dealing with high-dimensional sparse regression. The non-convex setting has only seen real progress recently. The first private SGD algorithm was given in [52], but only until the emergence of DP-SGD [1] with improved privacy composition do we see the real deployment of DP in deep learning systems. Since then, [4, 39] proposed Concentrated-DP and Rényi-DP which provides better composability for DP-SGD. [61] further improved the composition by considering sub-sampling in Rényi differential privacy. [31] proposed to use direct feedback alignment instead of backward propagation in DP-SGD. [43] suggested that a family of bounded activation functions, namely the tempered sigmoids, is more preferable than ReLU in DP-SGD. [22] used recurrent neural networks learned on auxiliary public data to adjust noise scales and to decide the update direction from the noisy gradient. [30] introduced a line-search module to privately choose among a pre-defined set of learning rates and decide whether to allocate more privacy budget. In addition, [30] showed that setting a smaller gradient clipping threshold C may impair utility when it causes too much information loss in the estimates. Besides, unlike randomized smoothing used in this work, reducing C does not affect the smoothness of loss landscapes and hence does not help tolerate noise.

Randomized Smoothing. Typically, a loss function can be viewed as a function of both model parameters and input data. In the machine learning community, several lines of works perform randomized smoothing over model parameter space as we do. To improve generalization, [6] proposed Entropy-SGD, which optimizes local entropy instead of the original loss function by estimating gradients with Langevin dynamics[62] and local entropy is designed to have a smooth energy landscape. With a similar purpose, [63] proposed SmoothOut, which injects noise into the model to smooth out sharp minima to obtain more robust models with flatter minima. [10] utilized parameter-space smoothing to improve convergence rates in nonsmooth convex optimizations.

7 Conclusion

In this work, we propose DPLIS, which improves the utility of DP-SGD by smoothing the learning loss function. We show both theoretically and empirically that DP-SGD with the smoothed loss not only reduces the performance variation across different runs but also achieves a better generalization bound. In addition, we show that the idea of smoothing learning loss can be easily extended to improve the utility of PATE.

Through our work, we hope to open up a new perspective for improving the utility of privacy-preserving deep learning, which is to tailor learning objective functions to differential privacy goals. For future work, we will investigate different instantiations of this overarching idea and study the applications of the proposed approach to medical data analysis.

8 Acknowledgement

This work is supported by National Natural Science Foundation of China (NSFC) under grant no. 61972448, DARPA contract #N66001-15-C-4066, the Center for Long-Term Cybersecurity, and Berkeley Deep Drive.

References

[1] M. Abadi, A. Chu, I. J. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, editors, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 308–318. ACM, 2016.

[2] R. Al-Rfou, M. Pickett, J. Snaider, Y.-h. Sung, B. Strope, and R. Kurzweil. Conversational contextual cues: The case of personalization and history for response ranking. *arXiv preprint arXiv:1606.00372*, 2016.

[3] G. Ateniese, L. V. Mancini, A. Spognardi, A. Villani, D. Vitali, and G. Felici. Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. *International Journal of Security and Networks*, 10(3):137–150, 2015.

[4] M. Bun and T. Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography Conference*, pages 635–658. Springer, 2016.

[5] N. Carlini, C. Liu, Ú. Erlingsson, J. Kos, and D. Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*, pages 267–284, 2019.

[6] P. Chaudhari, A. Choromanska, S. Soatto, Y. LeCun, C. Baldassi, C. Borgs, J. Chayes, L. Sagun, and R. Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124018, 2019.

[7] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(3), 2011.

[8] D. Chen, N. Yu, Y. Zhang, and M. Fritz. Gan-leaks: A taxonomy of membership inference attacks against gans. *arXiv preprint arXiv:1909.03935*, 2019.

[9] G. Chiachia, A. X. Falcão, N. Pinto, A. Rocha, and D. D. Cox. Learning person-specific representations from faces in the wild. *IEEE Trans. Inf. Forensics Secur.*, 9(12):2089–2099, 2014.

[10] J. C. Duchi, P. L. Bartlett, and M. J. Wainwright. Randomized smoothing for (parallel) stochastic optimization. In *Proceedings of the 51th IEEE Conference on Decision and Control, CDC 2012, December 10-13, 2012, Maui, HI, USA*, pages 5442–5444. IEEE, 2012.

[11] C. Dwork. Differential privacy. In M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, editors, *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, volume 4052 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2006.

[12] C. Dwork and J. Lei. Differential privacy and robust statistics. *Proceedings of the Annual ACM Symposium on Theory of Computing*, pages 371–380, 2009.

[13] C. Dwork and A. Roth. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3-4):211–407, 2014.

[14] C. Dwork, A. Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.

[15] W. N. Francis and H. Kucera. Brown corpus manual. *Letters to the Editor*, 5(2):7, 1979.

[16] M. Fredrikson, S. Jha, and T. Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333, 2015.

[17] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, pages 17–32, 2014.

[18] K. Ganju, Q. Wang, W. Yang, C. A. Gunter, and N. Borisov. Property inference attacks on fully connected neural networks using permutation invariant representations. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 619–633, 2018.

[19] S. Ghadimi and G. Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

[20] M. Hardt, B. Recht, and Y. Singer. Train faster, generalize better: Stability of stochastic gradient descent. In M. Balcan and K. Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1225–1234. JMLR.org, 2016.

[21] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016.

[22] J. Hong, H. Wang, Z. Wang, and J. Zhou. Learning model-based privacy protection under budget constraints. 2021.

[23] R. Iyengar, J. P. Near, D. Song, O. Thakkar, A. Thakurta, and L. Wang. Towards practical differentially private convex optimization. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 299–316. IEEE, 2019.

[24] P. Jain and A. G. Thakurta. (near) dimension independent risk bounds for differentially private learning. In *International Conference on Machine Learning*, pages 476–484, 2014.

[25] G. Kerrigan, D. Slack, and J. Tuyls. Differentially private language models benefit from public pre-training. *arXiv preprint arXiv:2009.05886*, 2020.

[26] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[27] D. Kifer, A. Smith, and A. Thakurta. Private convex empirical risk minimization and high-dimensional regression. In *Conference on Learning Theory*, pages 25–1, 2012.

[28] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[29] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

[30] J. Lee and D. Kifer. Concentrated differentially private gradient descent with adaptive per-iteration privacy budget. In Y. Guo and F. Farooq, editors, *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 1656–1665. ACM, 2018.

[31] J. Lee and D. Kifer. Differentially private deep learning with direct feedback alignment. *arXiv preprint arXiv:2010.03701*, 2020.

[32] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein. Visualizing the loss landscape of neural nets. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 6391–6401, 2018.

[33] Z. Li and Y. Zhang. Label-leaks: Membership inference attack with label. *arXiv preprint arXiv:2007.15528*, 2020.

[34] J. Liu and K. Talwar. Private selection from private candidates. In M. Charikar and E. Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 298–309. ACM, 2019.

[35] J. Liu, C. Zhang, et al. Distributed learning systems with first-order methods. *Foundations and Trends® in Databases*, 9(1):1–100, 2020.

[36] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[37] Y. Long, V. Bindschaedler, L. Wang, D. Bu, X. Wang, H. Tang, C. A. Gunter, and K. Chen. Understanding membership inferences on well-generalized learning models. *arXiv preprint arXiv:1802.04889*, 2018.

[38] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang. Learning differentially private recurrent language models. *arXiv preprint arXiv:1710.06963*, 2017.

[39] I. Mironov. Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 263–275. IEEE, 2017.

[40] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Ng. Reading digits in natural images with unsupervised feature learning. *NIPS*, 01 2011.

[41] N. Papernot, M. Abadi, U. Erlingsson, I. Goodfellow, and K. Talwar. Semi-supervised knowledge transfer for deep learning from private training data. *arXiv preprint arXiv:1610.05755*, 2016.

[42] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, and Ú. Erlingsson. Scalable private learning with pate. *arXiv preprint arXiv:1802.08908*, 2018.

[43] N. Papernot, A. Thakurta, S. Song, S. Chien, and Ú. Erlingsson. Tempered sigmoid activations for deep learning with differential privacy. *CoRR*, abs/2007.14191, 2020.

[44] N. Pinto, Z. Stone, T. E. Zickler, and D. D. Cox. Scaling up biologically-inspired computer vision: A case study in unconstrained face recognition on facebook. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2011, Colorado Springs, CO, USA, 20-25 June, 2011*, pages 35–42. IEEE Computer Society, 2011.

[45] Z. Qin, Y. Yang, T. Yu, I. Khalil, X. Xiao, and K. Ren. Heavy hitter estimation over set-valued data with local differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 192–203, 2016.

[46] M. A. Rahman, T. Rahman, R. Laganière, N. Mohammed, and Y. Wang. Membership inference attack against differentially private deep learning model. *Trans. Data Priv.*, 11(1):61–79, 2018.

[47] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[48] A. Salem, A. Bhattacharya, M. Backes, M. Fritz, and Y. Zhang. Updates-leak: Data set inference and reconstruction attacks in online learning. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*, pages 1291–1308, 2020.

[49] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. *arXiv preprint arXiv:1806.01246*, 2018.

[50] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2017.

[51] M. T. Smith, M. A. Álvarez, and N. D. Lawrence. Differentially private regression and classification with sparse gaussian processes. *CoRR*, abs/1909.09147, 2019.

[52] S. Song, K. Chaudhuri, and A. D. Sarwate. Stochastic gradient descent with differentially private updates. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 245–248. IEEE, 2013.

[53] K. Talwar, A. Thakurta, and L. Zhang. Private empirical risk minimization beyond the worst case: The effect of the constraint set geometry. *arXiv preprint arXiv:1411.5417*, 2014.

[54] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 2019.

[55] O. Thakkar, G. Andrew, and H. B. McMahan. Differentially private learning with adaptive clipping. *arXiv preprint arXiv:1905.03871*, 2019.

[56] A. G. Thakurta and A. Smith. Differentially private feature selection via stability arguments, and the robustness of the lasso. In *Conference on Learning Theory*, pages 819–850, 2013.

[57] B. Wang, Q. Gu, M. Boedihardjo, F. Barekat, and S. J. Osher. DP-LSSGD: A stochastic optimization method

to lift the utility in privacy-preserving ERM. *CoRR*, abs/1906.12056, 2019.

[58] L. Wang, B. Jayaraman, D. Evans, and Q. Gu. Efficient privacy-preserving nonconvex optimization. *CoRR*, abs/1910.13659, 2019.

[59] T. Wang and F. Kerschbaum. Robust and undetectable white-box watermarks for deep neural networks. *arXiv preprint arXiv:1910.14268*, 2019.

[60] T. Wang, Y. Zhang, and R. Jia. Improving robustness to model inversion attacks via mutual information regularization. *arXiv preprint arXiv:2009.05241*, 2020.

[61] Y.-X. Wang, B. Balle, and S. P. Kasiviswanathan. Subsampled rényi differential privacy and analytical moments accountant. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1226–1235. PMLR, 2019.

[62] M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient langevin dynamics. In L. Getoor and T. Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 681–688. Omnipress, 2011.

[63] W. Wen, Y. Wang, F. Yan, C. Xu, C. Wu, Y. Chen, and H. Li. Smoothout: Smoothing out sharp minima to improve generalization in deep learning. *arXiv preprint arXiv:1805.07898*, 2018.

[64] X. Wu, F. Li, A. Kumar, K. Chaudhuri, S. Jha, and J. Naughton. Bolt-on differential privacy for scalable stochastic gradient descent-based analytics. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1307–1322, 2017.

[65] Z. Yang, E.-C. Chang, and Z. Liang. Adversarial neural network inversion via auxiliary knowledge alignment. *arXiv preprint arXiv:1902.08552*, 2019.

[66] L. Yu, L. Liu, C. Pu, M. E. Gursoy, and S. Truex. Differentially private model publishing for deep learning. In *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*, pages 332–349. IEEE, 2019.

[67] J. Zhang, K. Zheng, W. Mou, and L. Wang. Efficient private erm for smooth objectives. *arXiv preprint arXiv:1703.09947*, 2017.

[68] Y. Zhang, R. Jia, H. Pei, W. Wang, B. Li, and D. Song. The secret revealer: generative model-inversion attacks against deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 253–261, 2020.

[69] Y. Zhu and Y.-X. Wang. Poission subsampled rényi differential privacy. volume 97 of *Proceedings of Machine Learning Research*, pages 7634–7642, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

Appendices

A Proof of Lemma 1

Proof. Let $\mu(\Delta)$ to be the density of a random variable $\Delta \sim \mathcal{N}(0, \sigma^2_{\text{smooth}}\mathbf{I})$, then we have

$$\begin{aligned}\mathcal{L}_{\text{smooth}}(\theta) &= \mathbb{E}_{\Delta\sim\mathcal{N}(0,\sigma^2_{\text{smooth}}\mathbf{I})}[\mathcal{L}(\theta+\Delta)] \\ &= \mathbb{E}_{\theta'\sim\mathcal{N}(\theta,\sigma^2_{\text{smooth}}\mathbf{I})}[\mathcal{L}(\theta')] \\ &= \int_{\theta'} \mathcal{L}(\theta')\mu(\theta-\theta')d\theta'.\end{aligned}$$

With Lemma 9(iii) from [10], given that $\mathcal{L}$ is L-Lipschitz with respect to the ℓ_2 norm, we have the gradient of $\mathcal{L}_{\text{smooth}}$ to be L/σ_{smooth}-Lipschitz continuous and therefore $\mathcal{L}_{\text{smooth}}$ is L/σ_{smooth}-smooth.

With Lemma 9(iv) from [10], we have

$$\begin{aligned}\nabla\mathcal{L}_{\text{smooth}}(\theta) =& \mathbb{E}_{\Delta\sim\mathcal{N}(0,\sigma^2_{\text{smooth}}\mathbf{I})}[\nabla\mathcal{L}(\theta+\Delta)] \\ &= \int_{\Delta} \left(\nabla\mathcal{L}(\theta+\Delta)\right)\mu(\Delta)d\Delta\end{aligned}$$

Thus when $\mathcal{L}$ is β-smooth, for any θ, θ', we have

$$\begin{aligned}&\|\nabla\mathcal{L}_{\text{smooth}}(\theta) - \nabla\mathcal{L}_{\text{smooth}}(\theta')\| \\ =&\|\int_{\Delta} \left(\nabla\mathcal{L}(\theta+\Delta)\right)\mu(\Delta)d\Delta - \int_{\Delta}\left(\nabla\mathcal{L}(\theta'+\Delta)\right)\mu(\Delta)d\Delta\| \\ =&\|\int_{\Delta}\left(\nabla\mathcal{L}(\theta+\Delta) - \nabla\mathcal{L}(\theta'+\Delta)\right)\mu(\Delta)d\Delta\| \\ \leq& \max_{\Delta}\|\nabla\mathcal{L}(\theta+\Delta) - \nabla\mathcal{L}(\theta'+\Delta)\| \\ \leq&\beta\|\theta-\theta'\|,\end{aligned}$$

which means $\mathcal{L}_{\text{smooth}}$ is at least β-smooth.

□

B Proof of Theorem 2

Proof. Since $\mathcal{L}$ is β-smooth,

$$\begin{aligned}\mathcal{L}(\theta_{t+1}) &\leq \mathcal{L}(\theta_t) + \langle\nabla\mathcal{L}(\theta_t), \theta_{t+1}-\theta_t\rangle + \frac{\beta}{2}\eta^2\|\nabla\mathcal{L}(\theta_t, x_t)\|^2 \\ &= \mathcal{L}(\theta_t) - \eta\langle\nabla\mathcal{L}(\theta_t), \nabla\mathcal{L}(\theta_t, x_t)\rangle + \frac{\beta}{2}\eta^2\|\nabla\mathcal{L}(\theta_t, x_t)\|^2\end{aligned}$$

Take expectation on both sides, we have

$$\begin{aligned}&\mathbb{E}[\mathcal{L}(\theta_{t+1})] - \mathbb{E}[\mathcal{L}(\theta_t)] \\ &\leq -\eta\|\nabla\mathcal{L}(\theta_t)\|^2 + \frac{\beta}{2}\eta^2\|\nabla\mathcal{L}(\theta_t)\|^2 + \frac{\beta}{2}\eta^2\sigma^2\end{aligned}$$

By summarizing the above equation through all time steps, we obtain the following inequality.

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\|\nabla\mathcal{L}(\theta_t)\|^2] \leq \frac{2\mathbb{E}[\mathcal{L}(\theta_1)] - 2\mathbb{E}[\mathcal{L}(\theta_{t+1})]}{\eta(2-\eta\beta)T} + \frac{\eta\beta\sigma^2}{(2-\eta\beta)}$$
$$\leq \frac{2\mathbb{E}[\mathcal{L}(\theta_1)] - 2\mathbb{E}[\mathcal{L}(\theta^*)]}{\eta(2-\eta\beta)T} + \frac{\eta\beta\sigma^2}{(2-\eta\beta)}$$

□

C Proof of Theorem 3

Proof. We introduce as tools the notion of uniform stability and a corresponding generalization bound from [20].

Definition 6 (Uniform Stability)**.** *A randomized learning algorithm $\mathcal{A}$ is s-uniformly stable if for all datasets D, D' of size n that differ in one sample, we have*

$$\mathbb{E}[\mathcal{L}(\theta_D, x) - \mathcal{L}(\theta_{D'}, x)] \leq s$$

for all sample x, where θ_D and $\theta_{D'}$ are respectively the final parameter learned from D and D' with $\mathcal{A}$ and the expectation is taken over the randomness of $\mathcal{A}$.

Theorem 4 (Generalization with Uniform Stability)**.** *Let $\mathcal{A}$ be s-uniformly stable. We have*

$$|\mathbb{E}[\hat{\mathcal{L}}(\theta) - \mathcal{L}(\theta)]| \leq s,$$

where θ is the final parameter learned from training data with $\mathcal{A}$ and the expectation is taken over both the randomness of $\mathcal{A}$ and the draw of training data.

We can now focus on bounding the uniform stability of DP-SGD, which can be then directly transferred into a bound for generalization gaps with Theorem 4.

Let $D = \{x_1, \ldots, x_N\}, D' = \{x'_1, \ldots, x'_N\}$ be two datasets of size n that differ in one sample(without loss of generality, we assume $x_1 \neq x'_1$)

Firstly, we analyze DP-SGD given realizations of the following randomness: Let $\theta_0 = \theta'_0$ be a realization of the random initialization of DP-SGD, $\{n_1, \ldots, n_T\} = \{n'_1, \ldots, n'_T\}$ be a realization of Gaussian noise from $\mathcal{N}(0, \sigma^2 C^2 \mathbf{I})$ for all T steps of DP-SGD and $\{B_1, \ldots, B_T\} = \{B'_1, \ldots, B'_T\}$ be a realization of indices of samples in all T steps.

We use θ_i and θ'_i to denote respectively the parameters after the i-th updates of DP-SGD on D and D', and use h_i to denote $\|\theta_i - \theta'_i\|$. Thus we have $h_0 = 0$.

In i-th step, if the one sample that differs in D and D' is not selected, we have

$$\begin{aligned} h_i \leq & h_{i-1} + \frac{\eta}{qn} \cdot \sum_{j\in B_i} \|\text{clip}\left(\nabla\mathcal{L}_{\text{train}}(\theta_{i-1}, x_j)\right) \\ & - \text{clip}\left(\nabla\mathcal{L}_{\text{train}}(\theta'_{i-1}, x_j)\right)\| \\ \leq & h_{i-1} + \frac{\eta}{qn}|B_i| \cdot \beta h_{i-1} \\ = & h_{i-1} \cdot (1+\eta\beta) \end{aligned}$$

where q is the sampling probability(i.e. $qn = |B_i|$ when using fixed-size batches) and the second inequality uses the condition that $\mathcal{L}_{\text{train}}$ is β-smooth(so that $\text{clip}\left(\nabla\mathcal{L}_{\text{train}}\right)$ is β-Lipschitz).

Similarly, if the one sample that differs in D and D' is selected, we have

$$\begin{aligned} h_i \leq & h_{i-1} + \frac{\eta}{qn} \cdot \sum_{j\in B_i} \|\text{clip}\left(\nabla\mathcal{L}_{\text{train}}(\theta_{i-1}, x_j)\right) \\ & - \text{clip}\left(\nabla\mathcal{L}_{\text{train}}(\theta'_{i-1}, x'_j)\right)\| \\ \leq & h_{i-1} + \frac{\eta}{qn}(|B_i| - 1) \cdot \beta h_{i-1} + \frac{\eta}{qn} \cdot 2C \\ \leq & h_{i-1} \cdot (1+\eta\beta) + \frac{2\eta C}{qn} \end{aligned}$$

Combining both cases with the inital condition $h_0 = 0$, we have

$$h_T \leq N_1 \cdot \frac{2\eta C}{qn} \cdot (1+\eta\beta)^{T-1},$$

where N_1 is the total number of steps that contains the different sample(i.e. x_1 and x'_1).

Since $\mathcal{L}(\theta, x)$ is L-Lipschitz for every x, we have

$$\begin{aligned} \mathbb{E}[\mathcal{L}(\theta_T, x) - \mathcal{L}(\theta'_T, x)] \leq & L\|\theta_T - \theta'_T\| \\ = & Lh_T \\ \leq & L \cdot N_1 \cdot \frac{2\eta C}{qn} \cdot (1+\eta\beta)^{T-1} \end{aligned}$$

By taking expectation on both side over all previously given realizations, we have

$$\begin{aligned} \mathbb{E}[\mathcal{L}(\theta_T, x) - \mathcal{L}(\theta'_T, x)] \leq & L \cdot \mathbb{E}[N_1] \cdot \frac{2\eta C}{qn} \cdot (1+\eta\beta)^{T-1} \\ = & L \cdot qT \cdot \frac{2\eta C}{qn} \cdot (1+\eta\beta)^{T-1} \\ = & \frac{2\eta C L T}{n} \cdot (1+\eta\beta)^{T-1} \end{aligned}$$

Since it holds for all D, D' of size n that differ in one sample, by Theorem 4, we have

$$|\mathbb{E}[\hat{\mathcal{L}}(\theta) - \mathcal{L}(\theta)]| \leq \frac{2\eta C L T}{n} \cdot (1+\eta\beta)^{T-1}.$$

□

D Experimental Setup: Evaluation of Smoothing for PATE

We evaluate PATE on SVHN[40] benchmark, which consists of 73257 images for training and 26032 images for testing. We preserve 10000 samples from its original test set for evaluating model performance and use the remaining 16032 samples as the source of public data.

Following [42], we use the labels produced by ensembles of 250 teachers published by [41]. We use Confident-GNMax Aggregator[42] to aggregate predictions of teachers. We fix $T = 300, \sigma_1 = 200$ and report test accuracy and privacy budgets corresponding to total number of label queries $\#\text{queries} = 4000, 8000, 12000, 16000$ with $\sigma_2 = 40, 80, 100$. In presenting privacy budgets, we fix δ to be 10^{-5} and present only the corresponding ε for simplicity. To highlight the utility derived from privacy-preserving labels rather than public data, student models in this section are trained in a supervised manner using public data with privacy-preserving labels.

Smoothing is applied to the learning loss as follows when training student models:

$$\mathcal{L}_{\text{smooth}}(\theta_{\text{student}}) = \frac{1}{K}\sum_{j=1}^{K}\mathcal{L}(\theta_{\text{student}} + \mathcal{N}(0, \sigma^2_{\text{smooth}}\mathbf{I})),$$

where K is number of smoothing samples and σ_{smooth} controls the degree of smoothing. We set $K = 10$ in all settings. We use a batch size of 100 and a learning rate of 0.01 that decays to 0.001 in the middle of training. In each setting, we train the student model for sufficiently long and report the highest test accuracy among training. The architecture of the student model is a CNN inherited from the official tutorial of tensorflow/privacy[2]. Table 13 contains other details of evaluation.

Table 13. Details of Evaluation of Smoothing for PATE

parameter	#queries	#labeled	ε	σ_{smooth}	P_{correct}
$\sigma_2 = 40$	**4000**	**1323**	**7.44**	**0.03**	**91.38%**
	8000	**2692**	**11.08**	**0.03**	**91.01%**
	12000	**4128**	**14.08**	**0.03**	**91.23%**
	16000	**5521**	**16.75**	**0.025**	**91.66%**
$\sigma_2 = 80$	**4000**	**1323**	**3.78**	**0.04**	**81.41%**
	8000	**2692**	**5.51**	**0.04**	**81.35%**
	12000	**4128**	**6.90**	**0.03**	**81.61%**
	16000	**5521**	**8.10**	**0.03**	**81.89%**
$\sigma_2 = 100$	**4000**	**1323**	**3.15**	**0.03**	**69.39%**
	8000	**2692**	**4.57**	**0.04**	**70.54%**
	12000	**4128**	**5.70**	**0.03**	**70.93%**
	16000	**5521**	**6.68**	**0.03**	**71.26%**

Notes

[1] https://github.com/yuxiangw/autodp
[2] https://github.com/tensorflow/privacy
[3] https://github.com/lukemelas/EfficientNet-PyTorch
[4] https://ic.unicamp.br/~chiachia/resources/pubfig83-aligned/
[5] https://github.com/ZhaoJ9014/face.evoLVe.PyTorch
[6] https://github.com/tomgoldstein/loss-landscape