# DPAdapter: Improving Differentially Private Deep Learning through Noise Tolerance Pre-training

Zihao Wang[1]*, Rui Zhu[1]*, Dongruo Zhou[1], Zhikun Zhang[3], John Mitchell[2],
Haixu Tang[1], and XiaoFeng Wang[1]
[1]*Indiana University Bloomington* [2]*Stanford University* [3]*Zhejiang University*

## Abstract

Recent developments have underscored the critical role of *differential privacy* (DP) in safeguarding individual data for training machine learning models. However, integrating DP oftentimes incurs significant model performance degradation due to the perturbation introduced into the training process, presenting a formidable challenge in the differentially private machine learning (DPML) field. To this end, several mitigative efforts have been proposed, typically revolving around formulating new DPML algorithms or relaxing DP definitions to harmonize with distinct contexts. In spite of these initiatives, the diminishment induced by DP on models, particularly large-scale models, remains substantial and thus, necessitates an innovative solution that adeptly circumnavigates the consequential impairment of model utility.

In response, we introduce DPAdapter, a pioneering technique designed to amplify the model performance of DPML algorithms by enhancing parameter robustness. The fundamental intuition behind this strategy is that models with robust parameters are inherently more resistant to the noise introduced by DP, thereby retaining better performance despite the perturbations. DPAdapter modifies and enhances the sharpness-aware minimization (SAM) technique, utilizing a two-batch strategy to provide a more accurate perturbation estimate and an efficient gradient descent, thereby improving parameter robustness against noise. Notably, DPAdapter can act as a plug-and-play component and be combined with existing DPML algorithms to further improve their performance. Our experiments show that DPAdapter vastly enhances state-of-the-art DPML algorithms, increasing average accuracy from 72.92% to 77.09% with a privacy budget of $\varepsilon = 4$.

## 1 Introduction

Recent years have witnessed an exponential growth in applications of *deep neural networks* (DNNs) to various domains [14, 22, 44]. However, DNN models trained with standard pipelines can be attacked by an adversary that seeks to reveal the data on which the model was trained. For example, Carlini et al. [9] show that adversaries can generate and detect text sequences from the training set of a large transformer language model, while Balle et al. [5] demonstrate that powerful adversaries can reconstruct images from the training set of a classifier. Alongside other results [8, 31, 32], these studies indicate that models trained on sensitive datasets present a significant privacy risk.

*Differential privacy* (DP) [16] has been the golden standard for effective control of the risks of exposing training examples, and has already been adopted by various machine learning tasks for protection [15, 30, 34, 50]. A differentially private algorithm is a randomized algorithm providing a formal guarantee that any single example in the training set only has a limited impact on the output distributions of the algorithm. The privacy guarantee, denoted ($\varepsilon$, $\delta$)-DP, is defined by two parameters ($\varepsilon$, $\delta$), which we refer to as the *privacy budget*. The smaller these two parameters are, the closer the output distributions between the training sets that differ by a single example, and therefore the more difficult it becomes for an adversary to infer whether any single data point is included during training.

**Challenges in Private Deep Learning.** *Differentially private stochastic gradient descent* (DP-SGD) [1] stands out as a prevalent DPML technique. DP-SGD modifies the conventional mini-batch gradient calculation used in SGD by incorporating a privatized version, whereby the gradient of each training sample is clipped to a maximum norm. Subsequently, Gaussian noise, proportional to the clipping norm, is introduced to the sum of the clipped gradients, masking the influence of any individual example on the sum. However, the incorporation of the DP noise often comes with a notable degradation in the performance of trained models [1, 47]. In response, efforts to mitigate the adverse impact of DP on model utility have been made [38, 47, 51, 56]. Nonetheless, the detriment imposed by DP noise on differentially private models remains significant.

*The first two authors contributed equally to this work.

**Our Solution.** To mitigate the adverse impact of DP on model utility, prior research has underscored that transfer learning [37] from public data markedly enhances the model performance of DPML algorithms [26]. In our research, we went a step further. Specifically, we observed that the deep learning training process is highly sensitive to noise, with even a small amount of noise leading to a substantial impact on performance. However, this issue can be mitigated by the model pre-trained to minimize parameter sensitivity to noise, which we call *parameter robustness*, so as to control DP's performance impact. Therefore, we designed a new technique that pre-trains a model with its parameters robust to noise to enhance the performance of the downstream model fine-turned with DPML on private data.

A technique that could serve this purpose is *sharpness-aware minimization* (SAM), which augments parameter robustness [17, 49, 55] for the purpose of enhancing a model's generality. This technique, however, turns out to be less effective in controlling the performance impact of DPML, since it is not designed for maximizing parameter robustness. Concretely, SAM adds the worst-case perturbation to parameters before computing gradients and later removes the perturbation after each round of parameter updates. For this purpose, it calculates both the perturbation and gradients on a small batch of training samples. A problem is that although a small batch could be enough for improving a model's generality, it is inadequate for making an accurate estimate of the worst-case perturbation, thereby rendering the model parameters less robust than they could be. To address this issue, we enhanced SAM for our purpose with a new technique called DPAdapter, which utilizes two batches of training instances, a large one for a more accurate estimate of the perturbation and a small one for effective gradient descent to ensure convergence. We further theoretically analyzed why this approach improves parameter robustness against noise, thereby reducing DPML's performance impact on the downstream model fine-tuning.

**Empiricaly Results.** We implemented DPAdapter and evaluated its performance using a model pre-trained on CIFAR-100, which was then fine-tuned for classification tasks on CIFAR-10, SVHN, and STL-10 datasets. In these experiments, we utilized DPAdapter as the pre-training method and the CIFAR-100 dataset as the public dataset for the pre-training. The downstream tasks involve fine-tuning the pre-trained model with DPML. Our experimental results consistently show a stable improvement in the accuracy achieved by the downstream tasks, compared with those fine-tuned from the models without robustness enhancement. For instance, when the privacy budget is set to $\varepsilon = 4$ and DPML configured to DP-SGD (a popular setting), DPAdapter elevates the average accuracy across three downstream tasks to 77.09%. By comparison, when utilizing the pre-trained models without DPAdapter, the accuracy is 72.92%, over 4% below that of our approach. Note that all existing DPML enhancements could only achieve a performance gain no more than 3% [48] when the privacy budget is set to $\varepsilon = 4$. Also, many of them cannot work together, since they all focus on the downstream fine-tuning step. Our approach, however, is designed for more generic protection, independent of tasks, and compatible with these existing solutions, given its focus on the pre-training step.

**Contributions.** Our key contributions are outlined below:

• *New technique*. We developed DPAdapter, a new technique for enhancing parameter robustness, which leads to a general solution significantly outperforming yet compatible with existing techniques for controlling DPML's negative impacts.

• *Theoretical understandings*. We theoretically analyzed our solution to justify its effectiveness, unveiling intrinsic relations among parameter robustness, transferability, and DPML's performance impacts. Our analysis leads to new insights about how a pre-trained model can be designed to maximize the benefits of DPML.

• *Extensive empirical studies*. We conducted comprehensive empirical studies of DPAdapter on various downstream tasks and DPML algorithms under different privacy budgets. Our results show that our approach effectively enhances the performance of the downstream tasks fine-tuned by DPML.

## 2 Background

### 2.1 Differentially Private Machine Learning

*Differential privacy* [16] (DP) serves as a mathematical structure intended to quantify the assurance of privacy within data analysis, providing insights into the preservation of privacy when an individual's data becomes part of a publicly analyzed dataset. This concept typically involves a privacy budget, represented by $\varepsilon$, where a larger value signifies augmented privacy protection, and an optional failure probability, $\delta$, defines the permissible deviation from impeccable privacy. Formally, we can define differential privacy as:

**Definition 2.1** ($(\varepsilon, \delta)$-DP)**.** *Given two neighboring datasets $D$ and $D'$ differing by one record, a mechanism $\mathcal{M}$ satisfies $(\varepsilon, \delta)$-differential privacy if*

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

*where $\varepsilon$ is the privacy budget, and $\delta$ is the failure probability.*

**Gaussian Mechanism.** There are several approaches for designing mechanisms that satisfy $(\varepsilon, \delta)$-differential privacy, among which the Gaussian mechanism is the most widely used one. It computes a function $f$ on the dataset $D$ in a differentially private manner, by adding to $f(D)$ a random noise. The magnitude of the noise depends on $\Delta_f$, the *global sensitivity* or the $\ell_2$ sensitivity of $f$. Such a mechanism $\mathcal{M}$ is given below:

$$\mathcal{M}(D) \quad = f(D) + \mathcal{N}\left(0, \Delta_f^2 \sigma^2 \mathbf{I}\right)$$

where $\Delta_f = \max_{(D,D'):D\simeq D'} ||f(D) - f(D')||_2$, $\mathcal{N}(0, \Delta_f^2\sigma^2\mathbf{I})$ denotes a multi-dimensional random variable sampled from the normal distribution with mean 0 and standard deviation $\Delta_f\sigma$, and $\sigma = \sqrt{2\ln\frac{1.25}{\delta}}/\varepsilon$.

**DP-SGD.** The integration of differential privacy into deep learning aims to build models that can learn from data without compromising the privacy of individuals within the dataset. Differentially private stochastic gradient descent (DP-SGD) [1] is the most widely used algorithm to enforce DP guarantee for the deep learning models. It adapts the standard SGD algorithm by introducing a few privacy-preserving modifications: *gradient clipping* and *noise addition*.

The gradient clipping operation aims to limit the sensitivity of each gradient, ensuring that a single data point does not unduly influence the model's learning process. Following gradient clipping, Gaussian noise is added to the clipped gradients before they are used to update the model parameters. This noise ensures that the exact values of the gradients—which could reveal sensitive information—are masked.

Our analysis is centered on gradient-level defenses, but it is crucial to note that methods like the Private Aggregation of Teacher Ensembles (PATE) [39] also show promise in balancing between privacy protection and model performance. Despite its potential, the applicability of PATE in federated learning [28] (which often involves continuously updated or streaming data) presents practical challenges, primarily due to the frequent necessity for retraining or re-aggregating teacher models. Federated learning serves as a clear example of a scenario where gradient-level defenses, such as DP-SGD, offer superior flexibility. DP-SGD naturally aligns with federated learning by seamlessly integrating privacy mechanisms into the model's training process, thus effectively managing real-time data updates. This inherent adaptability, combined with the challenges associated with implementing PATE in dynamic environments, highlights the reasons PATE-like strategies are not the focus of our analysis.

**Privacy Budget Composition.** On a broader spectrum, the overall privacy budget calculation of DP-SGD is established by demonstrating the privacy budget of each iteration under certain $(\varepsilon, \delta)$ values, followed by applying amplification by subsampling and composition throughout the iterations.

## 2.2 Adversarial Robustness and Model Parameter Robustness

**Adversarial Training (AT).** It is widely regarded as the most effective defense method against adversarial attacks [4, 6, 18, 40, 46]. AT essentially employs a "fight fire with fire" approach. It introduces adversarial examples into the training set, which are slight, carefully calculated modifications of the original inputs. These modifications are designed to mislead the model into making erroneous predictions. By learning from these adversarial examples, the model improves its ability to correctly classify such inputs in the future, thereby enhancing its overall resilience and robustness against adversarial attacks.

**Parameter Robustness.** It typically refers to the sensitivity of a model to small perturbations in its parameters. When discussing parameter robustness, we are interested in the model output (e.g., predictions on testing data) changes when its parameters are subjected to some level of noise.

Specifically, given a model $f$ parameterized by $\theta$ and a test sample $x$, the robustness of the model with respect to its parameters can be defined as:

$$\rho(f) = \max_{\boldsymbol{x},\theta,\Delta} |f_{\theta+\Delta}(\boldsymbol{x}) - f_\theta(\boldsymbol{x})|/\|\Delta\|_2$$

where $\rho(f)$ represents the maximum change in the model's output. $\Delta$ is the noise added to the model parameters $\theta$. $\|\cdot\|$ denotes a norm, for instance, the L2 norm.

This definition provides a measure of the potential variation in the model's output when its parameters undergo perturbation due to noise. A diminutive value of $\rho$ is indicative of pronounced parameter robustness. Subsequently, we aim to *enhance* the parameter robustness of $f$, denoted by the objective of reducing the value of $\rho(f)$.

**Robust Accuracy.** We resort to *robust accuracy* as an evaluation metric for parameter robustness in the classification paradigm; Given a specific model $f_\theta$, we introduce Gaussian noise, represented as $N(0, 0.1)$, uniformly across parameters spanning all layers. The resultant perturbed model is represented as $f_{\hat{\theta}}$. The model's performance, when faced with this perturbation, is then assessed. Consequently, the robust accuracy is computed over a dataset $D$ and is delineated as:

$$\text{Robust Accuracy} = \frac{1}{|D|}\sum_{x\in D} I(f_{\hat{\theta}}(x) = y)$$

where $I$ is the indicator function that returns 1 when the condition inside is true and 0 otherwise. A model exhibiting a higher robust accuracy indicates stronger parameter robustness, corresponding to a lower value of $\rho$.

The optimization of parameter robustness can be achieved through a technique known as *sharpness-aware minimization* (SAM) [17, 49, 55]. This method typically involves computing the worst-case perturbation on the parameters. The optimization then takes place on the model parameters subjected to this worst-case perturbation. The intention is to ensure that the model retains a significant portion of its performance, even when subjected to the worst-case perturbation. However, this technique is not designed to maximize parameter robustness but to achieve better generality. Consequently, it proves to be less effective in mitigating the performance impact of DPML (Section 5.2).

## 3 Problem Formulation and Key Observations

### 3.1 Problem Formulation

**Threat Model.** We consider a scenario where an attacker applies privacy attacks [5, 8, 9, 31, 32] to unveil the training data of a model. To defend against the attacks, the defender endeavors to build the model using differential privacy, which provably protects sensitive data while preserving its original utility. Additionally, the defender possesses a public dataset related to the private data, aligning with the standard assumption of transfer learning that the source and the target datasets follow similar but not identical distributions.

**Problem Definition.** Our aim is to build a pre-trained model that can be used for improving both the accuracy and privacy protection of the models trained for downstream tasks. Specifically, to build a pre-trained model that serves as a beneficial initialization for DP fine-tuning, we seek to answer two key questions: 1) what properties the pre-trained model should exhibit for the benefit of downstream models? and consequently 2) how the pre-training of the model should be executed to acquire these properties?

### 3.2 Motivation and Key Observations

**Motivation.** Our motivation to enhance the performance of the DPML algorithm using pre-trained models is rooted in our understanding of adversarial robustness training. The traditional adversarial training (AT) incrementally conditions the model to resist perturbations of its input. However, in the DPML algorithm, the noise is predominantly introduced into the gradients, or equivalently, into the model's parameters. Hence, we ask if there is any technique similar to AT that may condition the model to resist perturbations of its parameters. Obviously, if this can be achieved, the impact of the noise introduced by the DPML algorithm on the model performance could be minimized. It turns out, such an AT method has been proposed previously, referred to as the *sharpness aware minimization (SAM)*, which confers noise resilience to parameters Section 2.2. Based on the premise, we introduce two working hypotheses.

We first consider a specialized transfer learning scenario where the training datasets for the upstream and downstream tasks are i.i.d. (independent and identically distributed); for instance, each of these training datasets represent a random sub-sample of an overarching dataset. Under this scenario, we propose the first hypothesis:

**Hypothesis 1.** *Assuming the datasets A and B are randomly sampled from the same distribution, if a pre-trained model built from the dataset A possesses enhanced parameter robustness, the model fine-tuned on the dataset B using the DPML algorithm under a fixed privacy budget will yield better performance.*

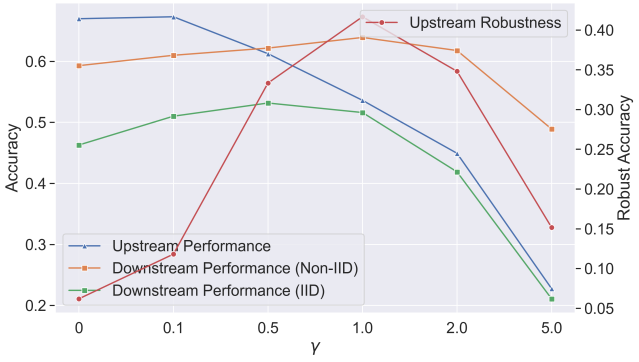


Figure 1: Impact of perturbation magnitude on AMP.

Next, we extend our hypothesis into a more general transfer learning scenario, where the upstream and downstream datasets are not identical but related, for example, when the upstream training data comes from ImageNet and the downstream data is sourced from CIFAR-10. Under this scenario, we have our second hypothesis.

**Hypothesis 2.** *Assuming the datasets A and B are sampled from two different but similar distributions, respectively, if a pre-trained model built from the dataset A exhibits strong parameter robustness, the model fine-tuned on the dataset B using the DPML algorithm under a fixed privacy budget will yield better performance.*

**Rationales.** Hypothesis 1 is quite intuitive. Drawing parallels from AT — which can counteract the effects of adversarial examples by introducing noise into the input — we anticipate that SAM can similarly mitigate the implications of the noise added to parameters by the DPML algorithm. Hypothesis 2, on the other hand, is inspired by prior work [42]. It has been observed that pre-trained models, when subjected to AT, can impart some degree of their acquired adversarial robustness to downstream models via transfer learning. This led us to a deeper contemplation: Can the parameter robustness garnered through SAM training also be transferred to downstream tasks through transfer learning? If this transfer of parameter robustness is indeed feasible, then, based on Hypothesis 1, it could potentially alleviate the performance degradation in downstream training using the DPML algorithm.

**Empirical Validation.** To evaluate our hypotheses, we utilized state-of-the-art Adversarial Model Perturbation (AMP) methodologies. Our models were exposed to an array of AMP intensities, producing pre-trained models that span a spectrum of parameter robustness. We then measured their parameter robustness by assessing the model accuracy on testing data after perturbation, and analyzed their performance in conjunction with DP-SGD.

Specifically, to validate Hypothesis 1, we examined a scenario where both upstream and downstream training data were sourced from the CIFAR-100 dataset, with the upstream

data constituting 90% of the CIFAR-100 training dataset and the downstream data constituting the remaining 10%. As depicted in Figure 1, the x-axis signifies the parameter $\gamma$ in AMP, which modulates the intensity of parameter robustness. By adjusting $\gamma$, we derived pre-trained models of various levels of robustness (indicated by the red line). This robustness peaks with increasing $\gamma$ and then starts to wane. Correspondingly, the green line, which illustrates the accuracy of the model's performance after downstream optimization using DP-SGD, displays an initial rise followed by a decline, mirroring the trajectory of parameter robustness. This result suggests that if the upstream and downstream training datasets both follow an i.i.d. distribution, the parameter robustness of upstream model is directly correlated with the performance of the downstream fine-tuned model.

For Hypothesis 2, we used the upstream training data comprising the same 90% of the CIFAR-100 dataset, while the downstream training data was entirely from the CIFAR-10 dataset. Still referencing Figure 1, the blue line indicates the performance (accuracy) of the downstream model after DP-SGD optimization. Consistent with the results above, the blue line showcases an initial increase in performance of the downstream model followed by a significant drop, which is highly consistent with the pattern of the parameter robustness. This result suggests that in a transfer learning scenario, the parameters robustness of an upstream pre-trained model can influence the performance of the downstream model trained using the DPML algorithm. Additionally, we observed a gradual decline in the accuracy of the upstream model with an increase in its parameter robustness, indicating a trade-off between the parameter robustness and the accuracy of the model. If the accuracy of the upstream model drops too significantly, it can detrimentally impact the performance of the downstream model. Taking these observation into account, we outline our design objectives below, aiming to illustrate our intent of devising a cost-effective strategy to build a pre-trained model that strikes a balance between two goals, i.e., to increase the parameter robustness while maintaining the high accuracy of the pre-trained model.

We further explore the transferability of parameter robustness in the context of transfer learning. In specific, we used CIFAR-100 as the upstream training dataset, and employed the CIFAR-10, SVHN, and STL-10 datasets, respectively for downstream training. We first trained the model on the upstream data with five different intensities of Adversarial Model Perturbation (AMP) characterized by the values $\gamma = 0, 0.1, 0.2, 0.5$, and 1.0. This procedure yielded five sets of pre-trained models, each with distinct robust accuracies. Subsequently, we performed transfer learning from each of these five distinct pre-trained models on the downstream data. Upon the completion of training, the robust accuracy of models on the downstream dataset was computed. Figure 2 shows the high between the robust accuracies of the pre-trained models (x-axis) and the robust accuracy of downstream models

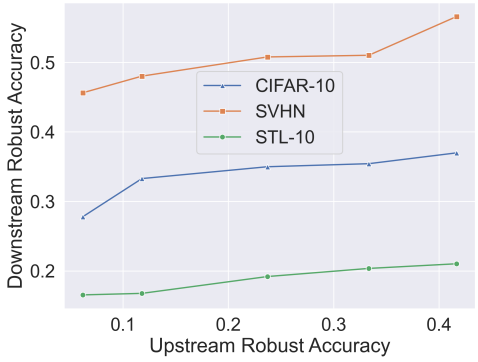


Figure 2: The relationship between the parameter robustness of the pretrained models and that of the models fine-tuned on the downstream tasks.

(y-axis). In general, it can be observed that the pre-trained models with higher robust accuracy tend to maintain their enhanced robust accuracy during the transfer learning for the downstream tasks. This observation underscores the inherent transferability of parameter robustness in a transfer learning context.

**Design Goals**. Based on the results from the toy examples as illustrated above, our objective is to build a pre-trained model that achieves two primary goals: robustness and effectiveness, which we elaborate below.

• *Robustness*. As observed earlier, we aim for our pre-trained model to exhibit strong parameter robustness (for instance, a smaller value of $\rho$). This ensures that when the downstream model employs the DPML algorithm, the influence on its performance is minimal.

• *Generality*. While striving for pronounced parameter robustness in the pre-trained model, it is imperative that the model's inherent performance remains commendable. Should we solely emphasize parameter robustness, the model may exhibit an extremely low value of $\rho$—indicating high robustness—but low accuracy. For instance, $f(x;\theta+\Delta) = f(x;\theta) \neq y$. As illustrated in Figure 1, when $\gamma$ exceeds 1, the pre-trained model's performance may deteriorate considerably. As a result, the DPML algorithm employed for downstream training might not inherit sufficient effective features from the pre-trained model, leading to a diminished outcome. Hence, it is equally pivotal for the pre-trained model to retain high performance as compared to enhance its parameter robustness.

## 4 DPAdapter

In essence, the design of DPAdapter is rooted in the two observations mentioned earlier. Our objective is for the pre-trained model to achieve higher parameter robustness while maintaining its performance. Given the observation that parameter robustness can be transferred when the model is passed down-

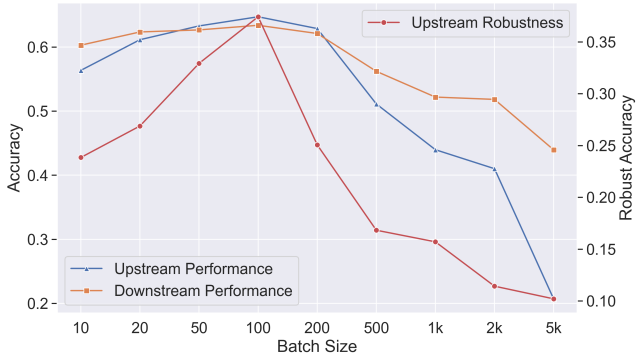


Figure 3: Impact of different batch sizes on AMP.

stream, the fine-tuning DPML algorithms can operate under the same privacy budget. This, in turn, minimizes the impact on performance.

## 4.1 Design Challenge

We observe that while AMP can enhance parameter robustness, leading to an improvement in the DP-SGD performance of downstream fine-tuned models, the extent of this robustness enhancement is quite limited. This, in turn, results in only a marginal performance boost. As we illustrated in Section 5.4 using AMP as an example, the effect of existing model parameter adversarial training on enhancing the parameter robustness of pre-trained models is not significant. Consequently, the impact on the performance of downstream DPML algorithms is also marginal. This is primarily because the existing methods aimed at improving parameter robustness were primarily designed to enhance model generalization. Since they were not explicitly designed for robustness enhancement, the gains in parameter robustness tend to be limited. While these methods can indeed be adjusted to prioritize robustness by increasing the perturbation magnitude at the expense of generalization, such a trade-off is often costly. As depicted in Figure 1, a high perturbation magnitude not only fails to bring much-added robustness but also makes the model challenging to converge. Given these challenges, we introduce DPAdapter, a novel approach to help pre-trained models achieve higher parameter robustness and minimize the adverse impact on performance when downstream models employ DPML algorithms.

## 4.2 Design Intuition

As highlighted in the "Challenge" section above, traditional methods of enhancing parameter robustness often result in a significant trade-off between parameter robustness and model performance. We've observed that these conventional methods of improving parameter robustness primarily leverage schemes akin to SGD for adversarial training. During a single optimization iteration, the batch used to compute the perturbation and the batch used to calculate the gradient direction for optimization are the same. However, the rationale behind this approach has not been extensively discussed. In fact, taking the AMP as an example, as depicted in Figure 3, when a large batch size is chosen, the model's training becomes suboptimal due to the utilization of this large batch. On the contrary, if the batch size is too small, computing the perturbation based on such limited data leads to inaccurate perturbations, subsequently compromising the robustness of the training. Note that a recent study suggests employing a linear scaling rule to adjust the learning rate based on the modified batch size [20]; specifically, the learning rate should be multiplied by $k$ when using a batch size of $kN$. Consequently, in this experiment, we also adhere to this setting to approximate the optimal learning rate for each batch size.

Based on these observations, we decided to decouple the batches used for computing perturbations and those for calculating parameter gradients during the optimization process. For computing perturbations, we chose a batch size larger than what is typically required for regular training. The reasoning is that perturbation computations don't necessarily benefit from being stochastic. Ideally, the entire dataset would be used to compute the perturbation. However, due to computational resource constraints, using the complete dataset for this purpose is challenging. Hence, we opt for a compromise by selecting a relatively larger batch size for the perturbation computations. In contrast, for batches used to compute model optimization gradients, we chose sizes that are conducive to model optimization, ensuring we avoid sizes that are excessively large or small that could impair model performance.

## 4.3 Methodology Overview

Figure 4 illustrates the overall workflow of DPAdapter. To efficiently achieve the parameter robustness of pre-trained models, DPAdapter employs a Min-Max optimization, iterating over four steps of model updating as presented below.

**Step 1: Warming-up.** The pre-trained model initiates with a warming-up phase using standard training techniques to ensure the accuracy (ACC) meets the targeted benchmark. We refer to the model post-warming-up as $f_\theta$.

**Step 2: Worst-case Model Perturbation.** We select an enlarged batch of training data, denoted as $\mathcal{B}_1$. We then compute the direction of the gradient that maximizes the loss of this batch under the current model parameters. Subsequently, we perturb the model parameters along this direction with the intention of impairing the model's predictive capability on this large batch of data when subjected to such perturbation. We denote the perturbation as $\Delta$, and the model after perturbation as $f_{\theta+\Delta}$.

**Step 3: Update on the Perturbed Model.** To enhance the model's resilience to perturbations in its parameters (param-

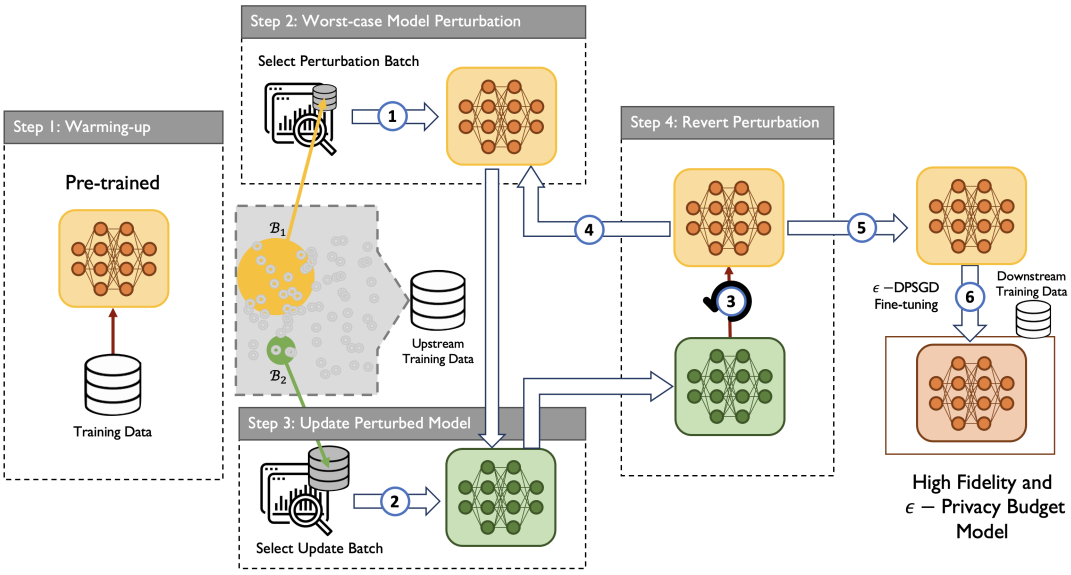


Figure 4: Overview of the DPAdapter approach.

eter robustness), we optimize the perturbed model with the goal of maintaining good performance even when the model is subjected to such perturbations. We select a standard-sized batch of training data, represented as $\mathcal{B}_2$. We then perform regular training using stochastic gradient descent (SGD) on $\mathcal{B}_2$. The updated model is denoted as $f_{\theta+\Delta+\alpha}$, where $\alpha$ represents the weight update resulting from the SGD.

**Step 4: Revert Perturbation.** In Step 3, we have already enhanced the model's resilience to parameter perturbations. As a result, there's no longer a need to retain the noise applied to the parameters in Step 2, as doing so would only degrade our model's accuracy. Therefore, in this step, we revert the update of the model weight made in Step 2: the final model is represented as $f_{\theta+\Delta+\alpha-\Delta} = f_{\theta+\alpha}$.

Our workflow operates iteratively. Upon completion of Step 4, the process loops back to Step 1. In each iteration, distinct batches of training data are randomly selected for both $\mathcal{B}_1$ and $\mathcal{B}_2$. The algorithm takes the total number of iterations as an input, ensuring that training concludes after the specified number of iterations.

**Remark.** It is important to highlight the distinction between our approach and conventional techniques aimed at enhancing a model's generality by improving parameter robustness, such as AMP [55]. Specifically, AMP optimizes using the same $\mathcal{B}_2$ and $\mathcal{B}_1$.

The key innovation in our method lies in the strategic batch selection during two different optimization processes: the worst-case perturbation computation and the SGD gradient computation. By ensuring that the batch size for $\mathcal{B}_1$ is relatively large, we obtain a more precise and general perturbation. In contrast, a standard batch size is used for $\mathcal{B}_2$ to facilitate optimal training with SGD. Such nuanced improvements enable our pretrained model, DPAdapter, to achieve heightened parameter robustness, resulting in significant enhancements when downstream DPSGD is applied.

## 4.4 Design Details

We now delve into a detailed presentation of the DPAdapter algorithm, focusing primarily on Steps 2 and 3. These steps are comparatively more complex: while Step 1 involves standard training and Step 4 simply involves a subtraction to remove noise from the parameters. Notably, it's Steps 2 and 3 that distinctly set DPAdapter apart from other SAM algorithms.

First, like any typical SAM algorithm, we define a *norm ball* $\mathbf{B}(\boldsymbol{\mu};\gamma)$, as a region around a given point $\boldsymbol{\mu}$ in the parameter space $\Theta$ (i.e., $\boldsymbol{\mu} \in \Theta$) with the radius of $\gamma$ ($\gamma \geq 0$):

$$\mathbf{B}(\boldsymbol{\mu};\gamma) := \{\boldsymbol{\theta} \in \Theta : \|\boldsymbol{\theta} - \boldsymbol{\mu}\| \leq \gamma\} \tag{1}$$

Recall that a typical SAM loss is designed upon the norm ball $\mathbf{B}(\theta;\gamma)$:

$$\mathcal{L}_{\text{SAM}}(\boldsymbol{\theta}) := \max_{\Delta \in \mathbf{B}(\mathbf{0};\gamma)} \frac{1}{|\mathcal{D}|} \sum_{(\boldsymbol{x},\boldsymbol{y}) \in \mathcal{D}} \ell(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\theta}+\Delta) \tag{2}$$

where $\mathcal{D}$ represent the training dataset. The training loss, denoted by $\ell$, varies based on the task at hand: for classification tasks, it is typically the cross-entropy, while for regression tasks, it is often the least squares. The perturbation $\Delta$ is the worst-case model perturbation

**Worst-case Model Perturbation.** In this section, we detail the computation of $\Delta$ used to obtain $f_{\theta+\Delta}$ in Step 2. The worst-case model perturbation, $\Delta$, is strategically designed to induce the model parameters $\theta$ to achieve the fastest reduction in loss

on the dataset:

$$\Delta_{\mathcal{B}_1} = \arg\max_{\Delta \in \mathbf{B}(\boldsymbol{\theta};\gamma)} \frac{1}{|\mathcal{B}_1|} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{B}_1} \ell(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\theta}+\Delta) \tag{3}$$

Recall that in DPAdapter, $\mathcal{B}_1$ denote the batch of data use as to compute the worst-case model perturbation. $\gamma$ denotes the radius of the norm ball and acts as a hyperparameter that governs the degree of parameter robustness training. The robustness of the model's parameters is directly influenced by this radius: a larger $\gamma$ implies that the model can tolerate more noise in the parameter space, as determined by DPAdapter. Consequently, a greater $\gamma$ results in a model with enhanced parameter robustness.

**Update Perturbed Model.** In this section, we detail the computation of $\alpha$ used to obtain $f_{\theta+\Delta+\alpha}$ in Step 3. Since in practice, the optimization is typically carried out using mini-batches. In this case, the $\mathcal{L}_{\text{DPAdapter}}$ can be approximated using a mini-batch $\mathcal{B}_2$:

$$\begin{aligned}\mathcal{L}_{\text{DPAdapter}}(\boldsymbol{\theta}) &\approx \max_{\Delta_{\mathcal{B}_1}\in\mathbf{B}(\mathbf{0};\gamma)} \frac{1}{|\mathcal{B}_2|} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{B}_2} \ell(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\theta}+\Delta_{\mathcal{B}_1}) \\ &:= \mathcal{J}_{\text{DPAdapter},\mathcal{B}_1,\mathcal{B}_2}(\boldsymbol{\theta})\end{aligned} \tag{4}$$

Hence, at this point, the model's update is given by

$$\alpha = \eta_2 \nabla \mathcal{J}_{\text{DPAdapter},\mathcal{B}_1,\mathcal{B}_2}(\boldsymbol{\theta}) \tag{5}$$

where $\eta_2$ represents the optimization step size. It is crucial to highlight the primary distinction between DPAdapter and other SAM algorithms, which lies in Equation 3 and Equation 4. In other SAM algorithms, such as those cited in [17, 49, 55], the batch used to compute the worst-case perturbation, denoted as $\mathcal{B}_1$, is the same as the batch utilized for computing the update. We argue that by separately selecting $\mathcal{B}_1$ and $\mathcal{B}_2$, the trained model can achieve a balance between parameter robustness and overall model performance in Section 5.4.

The specifics of the DPAdapter algorithm can be found in Algorithm 1. Importantly, in alignment with other SAM algorithms such as [17, 49, 55], the finalized model utilizes the parameters denoted as $\boldsymbol{\theta}^*_{\text{DPAdapter}}$. These parameters are employed for model inference (or prediction) without the need for additional perturbations.

As mentioned earlier, the training process of DPAdapter is divided into two primary stages: The perturbation computation and the update computation.

• **Perturbation Computation**. Line 5 to Line 9 in Algorithm 1. This phase is responsible for calculating the worst-case perturbation, $\Delta_{\mathcal{B}_1}$, based on the prevailing conditions.

• **Update Computation**. Line 10 to Line 12 in Algorithm 1. In this phase, the $\Delta_{\mathcal{B}_1}$ derived from the previous perturbation computation step is used to determine the parameter optimization gradient, $\nabla \mathcal{J}_{\text{DPAdapter},\mathcal{B}_1,\mathcal{B}_2}$. Following this, the model parameters are updated.

**Algorithm 1** DPAdapter Training

**Require:** Training set $\mathcal{D} = \{(\boldsymbol{x},\boldsymbol{y})\}$, perturbation batch size $m_1$, update batch size $m_2$, loss function $\ell$, perturbation computation learning rate $\eta_1$, update learning rate $\eta_2$, norm ball radius $\gamma$, number of update iteration $K$.

1: **while** $k < K$ **do**
2: Draw $\mathcal{B}_1 = \{(\boldsymbol{x}_i,\boldsymbol{y}_i)\}_{i=1}^{m_1}$ from training set $\mathcal{D}$
3: Draw $\mathcal{B}_2 = \{(\boldsymbol{x}_i,\boldsymbol{y}_i)\}_{i=1}^{m_2}$ from training set $\mathcal{D}$
4: Set perturbation: $\Delta_{\mathcal{B}_1} \leftarrow \mathbf{0}$
5: Calculate the perturbation:
$\Delta_{\mathcal{B}_1} \leftarrow \eta_1 \arg\max_{\Delta\in\mathbf{B}(\boldsymbol{\theta}_k;\gamma)} \frac{1}{|m_1|}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{B}_1} \ell(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\theta}_k+\Delta)$
6: **if** $\|\Delta_{\mathcal{B}_1}\|_2 > \gamma$ **then**
7: Normalize perturbation: $\Delta_{\mathcal{B}_1} \leftarrow \gamma\Delta_{\mathcal{B}_1}/\|\Delta_{\mathcal{B}_1}\|_2$
8: **end if**
9: Add the perturbation to parameters:
$\boldsymbol{\theta}_k \leftarrow \boldsymbol{\theta}_k + \Delta_{\mathcal{B}_1}$
10: Compute update gradient:
$\nabla \mathcal{J}_{\text{DPAdapter},\mathcal{B}_1,\mathcal{B}_2} \leftarrow \sum_{i=1}^{|\mathcal{B}_2|} \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{x}_i,\boldsymbol{y}_i;\boldsymbol{\theta}_k)/m_2$
11: Update on the perturbed model :
$\boldsymbol{\theta}_{k+1} \leftarrow \boldsymbol{\theta}_k - \eta_2 \nabla \mathcal{J}_{\text{DPAdapter},\mathcal{B}_1,\mathcal{B}_2}$
12: Revert perturbation: $\boldsymbol{\theta}_{k+1} \leftarrow \boldsymbol{\theta}_{k+1} - \Delta_{\mathcal{B}_1}$
13: Increment iteration: $k \leftarrow k+1$
14: **end while**

Training can be executed for a specified number of epochs $K$, or it can continue until the parameter loss stabilizes, which indicates the completion of the training process.

## 4.5 Theoretical Analyses

In this section, we aim to provide a theoretical explanation of why DPAdapter achieves improvements in the DPML algorithm. Specifically, our theoretical examination revolves around two key points:

1. Understanding the rationale behind the separate selection of $\mathcal{B}_1$ and $\mathcal{B}_2$, particularly emphasizing why the batch size of $\mathcal{B}_2$ should be large.

2. Elucidating how enhancing the robustness of pretrained parameters can ensure superior performance in downstream training, given the same privacy budget.

The first point is theoretically proven in Theorem 1, while the second is validated in Theorem 2.

To give a basic idea, through Theorem 1, we aim to theoretically demonstrate that, upon selecting an appropriate $\mathcal{B}_1$ conducive to optimization and holding it constant, we can bound the difference between the actual loss and the maximum lower bound (Infimum) of the loss. Essentially, as the magnitude of $\mathcal{B}_2$ increases, the upper bound on the expected convergence rate of the model obtained through DPAdapter also increases.

In theorem 2, we prove that given a model training by DP-SGD, the expected difference between the model's loss and

the maximum lower bound of the achievable loss (termed as DP-SGD's performance) is positively proportional to $\rho$. Specifically, a lower value of $\rho$ (indicating stronger parameter robustness) in the model initial stage, results in a lower loss when optimized using DP-SGD.

Next, we delineate the setup for our theoretical framework. We first analyze the convergence behavior of our proposed algorithm. We show that by using a different update batch (where $|\mathcal{B}_1| \neq |\mathcal{B}_2|$). We summarize our algorithm as follows. Let $L_i(\boldsymbol{\theta}) := \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i)$. For any batch of the data $\mathcal{S} \subseteq \mathcal{D}$, let $L_{\mathcal{S}}$ be the average of $L_i, i \in \mathcal{S}$. Then our algorithm is as follows. Starting from the initial parameter $\boldsymbol{\theta}_0$, for each iteration $t$, we have

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_2 \cdot \nabla L_{\mathcal{B}_2}(\boldsymbol{\theta}_t + \Delta_t),\ \Delta_t := \eta_1 \cdot \nabla L_{\mathcal{B}_1}(\boldsymbol{\theta}_t). \quad (6)$$

Equation 6 is essentially the same as Algorithm 1 in Section 4.3 without the perturbation normalization step (line 6-8 in Algorithm 1). Meanwhile, we need the following assumptions.

**Assumption 1.** *$f$ is $\beta$-smooth w.r.t. $\boldsymbol{\theta}$, i.e., $\|\nabla f_{\boldsymbol{\theta}}(\boldsymbol{x}) - \nabla f_{\boldsymbol{\theta}'}(\boldsymbol{x})\|_2 \leq \beta \|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2$.*

**Assumption 2.** *$\ell$ is $\beta_1$-Lipschitz continuous w.r.t. $\boldsymbol{x}$ i.e., $|\ell(\boldsymbol{x}, \boldsymbol{y}) - \ell(\boldsymbol{x}', \boldsymbol{y})| \leq \beta_1 |\boldsymbol{x} - \boldsymbol{x}'|$.*

**Assumption 3.** *$\ell$ is $\beta_2$-smooth w.r.t. $\boldsymbol{x}$, i.e., $|\nabla_{\boldsymbol{x}} \ell(\boldsymbol{x}, \boldsymbol{y}) - \nabla_{\boldsymbol{x}} \ell(\boldsymbol{x}', \boldsymbol{y})| \leq \beta_2 |\boldsymbol{x} - \boldsymbol{x}'|$.*

**Assumption 4.** *$\nabla L_i$ is $\hat{\sigma}^2$-variance bounded gradient, i.e., $\mathbb{E}_i \|\nabla L_i(\boldsymbol{\theta}) - \nabla L_{\mathcal{D}}(\boldsymbol{\theta})\|_2^2 \leq \hat{\sigma}^2$. Here $\mathbb{E}_i$ denotes the expectation over $i \in 1, \ldots, n$.*

**Assumption 5.** *$L_{\mathcal{D}}$ satisfies the $\mu$-PL-condition, i.e., $\|\nabla L_{\mathcal{D}}(\boldsymbol{\theta})\|_2^2 \geq 1/\mu \cdot (L_{\mathcal{D}}(\boldsymbol{\theta}) - \inf_{\hat{\boldsymbol{\theta}}} L_{\mathcal{D}}(\hat{\boldsymbol{\theta}}))$.*

We also formally define the parameter robustness in Section 2.2 as follows:

$$\rho(f) = \max_{\boldsymbol{x}, \boldsymbol{\theta}, \Delta} |f_{\theta+\Delta}(\boldsymbol{x}) - f_{\theta}(\boldsymbol{x})| / \|\Delta\|_2.$$

For simplicity, we use $\rho := \rho(f)$ with a slight abuse of notation. Then we have the following theorem.

**Theorem 1** (Informal)**.** *With proper selection of parameters, our algorithm enjoys the following gradient norm bound:*

$$\frac{1}{T} \sum_{t=1}^{T} \mathbb{E}(L_{\mathcal{D}}(\boldsymbol{\theta}_t) - \inf_{\hat{\boldsymbol{\theta}}} L_{\mathcal{D}}(\hat{\boldsymbol{\theta}})) \leq \mu \cdot \left( \frac{L_{\mathcal{D}}(\boldsymbol{\theta}_0)}{T} + \frac{\hat{\sigma}^2}{16\hat{\beta}} (1/|\mathcal{B}_2| + 1/|\mathcal{B}_1|) \right),$$

*where $\hat{\beta} = (\rho^2 \beta_2 + \beta\beta_1)$.*

We note that the significance of increasing the perturbation batch size, $|\mathcal{B}_1|$, directly influences the gradient norm bound and, consequently, the convergence rate of our algorithm. Elevating $|\mathcal{B}_1|$ effectively lowers this bound by diminishing the noise-to-signal ratio in our gradient estimations, leading to more precise and efficient optimization steps. This strategy is particularly crucial within the context of differential privacy, where striking a balance between privacy protection and algorithmic efficacy is paramount. By judiciously selecting a larger $|\mathcal{B}_1|$, we enable a smoother and more accurate convergence process, highlighting the importance of careful batch size configuration in achieving optimal learning outcomes under privacy constraints.

The detailed theorem and proof are deferred to Appendix A.1. Theorem 1 suggests that selecting a large perturbation batch size $|\mathcal{B}_1|$ makes our algorithm have a better convergence rate, indicating better performance while using a large batch of perturbations $|\mathcal{B}_1|$.

We have another analysis of the noise tolerance and the DP-SGD performance. Then we have our theorem.

**Theorem 2** (Informal)**.** *Given $(\varepsilon, \delta)$, with proper selection of parameters, let $\boldsymbol{\theta}_{out}$ be the final output of DP-SGD, then DP-SGD is $(\varepsilon, \delta)$-DP and enjoys the following utility bound:*

$$\mathbb{E}(L_{\mathcal{D}}(\boldsymbol{\theta}_{out}) - \inf_{\boldsymbol{\theta}} L_{\mathcal{D}}(\boldsymbol{\theta})) \leq C \cdot \frac{\rho \sqrt{\rho^2 \beta_2 + \beta\beta_1}}{|\mathcal{D}|\varepsilon},$$

*where $C = c \cdot \mu\beta_1 \sqrt{d \log(|\mathcal{D}|/\delta) \log(1/\delta) L_{\mathcal{D}}(\boldsymbol{\theta}_0)}$, $c$ is some positive constant, $d$ is the dimension of parameter $\boldsymbol{\theta}$.*

The detailed algorithm, theorem, and proof are deferred to Appendix A.2. Theorem 2 suggests that the parameter robustness $\rho$ indeed affects the final utility.

# 5 Evaluation

## 5.1 Experimental Setup

**Datasets**. We use the following five datasets in our evaluation.

• *CIFAR-10* [25]: This dataset contains 50,000 training images and 10,000 testing images. Each image has a size of $32 \times 32 \times 3$ and belongs to one of 10 classes.

• *CIFAR-100* [25]: This dataset contains 50,000 training images and 10,000 testing images. Each image has a size of $32 \times 32 \times 3$ and belongs to one of 100 classes.

• *SVHN* [36]: In this dataset, each image represents a digit from the house numbers in Google Street View. The size of each image is $32 \times 32 \times 3$. Each image belongs to one of the 10 digits. This dataset has 73,257 training images and 26,032 testing images.

• *STL-10* [13]: This dataset contains 5,000 training images and 8,000 testing images. Each image has a size of $96 \times 96 \times 3$ and belongs to one of 10 classes.

• *Tiny ImageNet* [27]: This dataset contains 100,000 training images and 10,000 testing images. Each image has a size of $64 \times 64 \times 3$ and belongs to one of 200 classes.

In this paper, CIFAR-100 and Tiny ImageNet serves as a public dataset used for pre-training, while CIFAR-10, SVHN, and STL-10 serves as the private data/tasks.

**Pre-training Procedure**. In our experiments, we utilize CIFAR-100 as our pre-training dataset and employ DPAdapter to train a ResNet20 model [22] as the pre-trained model. We compute the mean and standard deviation on the training set to normalize the input images. We adopt cross-entropy as the loss function $\ell$ and utilize the SGD optimizer with momentum, incorporating a step-wise learning rate decay. Specifically, the model is trained for $K = 1,000$ epochs, with the outer learning rate $\eta_2$ initialized at 0.1 and divided by 10 after 500 and 750 epochs. The momentum is set to 0.9, and the weight decay is set to $1 \times 10^{-4}$. The inner batch size $m_1$ is set to 5,000, with the inner learning rate $\eta_1$ set to 1.0; the outer batch size $m_2$ is set to 50. For perturbation magnitude, we adopt $\gamma = 2.0$ by default.

**Fine-tuning Procedure**. Given a pre-trained model, we utilize it to train private downstream classifiers for the remaining three datasets: CIFAR-10, SVHN, and STL-10. We employ the parameters of the pre-trained model as the initial parameters of the downstream classifier, which is then trained on the downstream dataset.

We use Rényi DP to accumulate the overall privacy budget and precompute the required noise scale ($\sigma$ in DP-SGD) numerically [1, 35]. We maintain $\delta = 10^{-5}$ and utilize different privacy budgets: $\varepsilon = \{1, 4\}$. The clipping threshold for all algorithms is fixed at 4, except when an algorithm employs special clipping strategies. The cross-entropy loss function and the DP-SGD optimizer with momentum are adopted when training a downstream classifier. The model is fine-tuned for 100 epochs, with the learning rate initialized at 0.01 and momentum set to 0.9. The batch size is configured to 256.

**DPML Algorithms.** According to the different optimization methods that the downstream model could adopt, we consider three types of DP algorithms in addition to vanilla DP-SGD:

• *GEP* [52]: Yu et al. observed that in vanilla DP-SGD, the amount of noise increases with the model size and proposed a solution, GEP [52], to reduce the gradient dimension before adding noise. GEP first calculates an anchor subspace, which contains some gradients of public data, using the power method. Subsequently, it projects the gradient of private data into the anchor subspace, resulting in a low-dimensional gradient embedding and a small-norm residual gradient. These two components are independently processed with the DP mechanism and then combined to update the original weight.

• *AdpAlloc* [53]: It proposes a dynamic noise-adding mechanism, eschewing the practice of maintaining a constant noise multiplier $\sigma$ throughout every training epoch in vanilla DP-SGD. Instead of utilizing a static variance in the Gaussian mechanism, it replaces it with a function of the epoch: $M(d) = f(d) + \mathbf{N}(0, S_f^2 \cdot \sigma_t^2)$, where the value of $\sigma_t$ is contingent upon the final privacy budget, epoch, and schedule function. The schedule function delineates the adjustment of the noise scale throughout training. Yu et al. proposed several pre-defined schedules. For our evaluation, we select *Exponential Decay*, which demonstrated the best average performance in [53]. The mathematical expression for *Exponential Decay* is $\sigma_t = \sigma_0 e^{-kt}$, where $k (k > 0)$ represents the decay rate and $\sigma_0$ denotes the initial noise scale.

• *AdpClip* [2]: An adaptive clipping threshold mechanism is utilized, setting the clip threshold to a specified quantile of the update norm distribution at each epoch. Formally, the clipping threshold $C_t$ in epoch $t$ can be computed as $C_t = C_{t-1} \cdot \exp\left(-\eta_C(\overline{b} - \gamma)\right)$, where $\gamma \in [0, 1]$ is a quantile to be matched, $\overline{b} \triangleq \frac{1}{m} \sum_{i \in [m]} \mathbb{I}_{x_i \leq C}$ represents the empirical fraction of samples with a value at most $C$, and $\eta_C$ is the learning rate with a default value of 0.2, as indicated in [2].

## 5.2 Effectiveness of DPAdapter

In this section, we empirically validate the overall effectiveness of DPAdapter and conduct ablation studies to illustrate the effectiveness of each component.

**Setup**. We conduct experiments using four different DPML algorithms across three distinct private downstream tasks (CIFAR-10, SVHN, and STL-10), under four different pre-training settings: (i) From Scratch: utilizing randomly initialized weight values (i.e., no pre-training); (ii) Standard Pre-training [15, 26]: conducting pre-training on the CIFAR-100 dataset using standard training procedures; (iii) Vanilla Sharpness-Aware Minimization (SAM): pre-training with AMP [55], employing the optimal perturbation magnitude $\gamma = 1.0$ and the optimal batch size $m = 100$; (iv) DPAdapter: pre-training with the proposed DPAdapter, using hyper-parameters described in Section 5.1.

**Observations**. Table 1 illustrates the downstream accuracy for various settings. In general, we observe that the proposed DPAdapter substantially enhances the downstream accuracy in all settings. For instance, when the privacy budget is set to $\varepsilon = 1$ and the DPML algorithm is configured to vanilla DP-SGD, DPAdapter elevates the average accuracy across three downstream datasets to 61.42%. In contrast, when utilizing models pre-trained normally, the accuracy is only 56.95%, marking an improvement of over 4%. When the DPML algorithm is AdpClip, DPAdapter boosts the accuracy from 42.67% to 52.54%, representing a near 10% improvement.

We note that, compared to training from scratch, a normally pre-trained model consistently yields an improvement in downstream accuracy. This finding aligns with previous work which noted that transfer learning from public data significantly enhances the accuracy of DPML algorithms [26]. Further, we note that employing vanilla SAM to provide pa-

Table 1: Comparison of the performance of training from scratch, standard pre-training, vanilla SAM, and DPAdapter across various downstream tasks under different privacy budgets.

| DPML Algorithms | Upstream Training Method | CIFAR10 | | SVHN | | STL10 | | Average | |
|---|---|---|---|---|---|---|---|---|---|
| | | $\varepsilon = 1$ | $\varepsilon = 4$ | $\varepsilon = 1$ | $\varepsilon = 4$ | $\varepsilon = 1$ | $\varepsilon = 4$ | $\varepsilon = 1$ | $\varepsilon = 4$ |
| DP-SGD | None (Scratch) [1] | 0.4288 | 0.5070 | 0.7194 | 0.8380 | 0.2831 | 0.3370 | 0.4771 | 0.5607 |
| | Standard Pre-training [15, 26] | 0.5928 | 0.7210 | 0.7822 | 0.8970 | 0.3282 | 0.5695 | 0.5677 | 0.7292 |
| | Vanilla SAM (Ours) | 0.6216 | 0.7650 | 0.8042 | 0.9014 | 0.3625 | 0.6212 | 0.5961 | 0.7625 |
| | DPAdapter (Ours) | **0.6416** | **0.7746** | **0.8058** | **0.9018** | **0.3951** | **0.6364** | **0.6142** | **0.7709** |
| AdpClip | None (Scratch) [2] | 0.3738 | 0.5348 | 0.6258 | 0.8294 | 0.2270 | 0.3543 | 0.4089 | 0.5728 |
| | Standard Pre-training [15, 26] | 0.4198 | 0.6780 | 0.6196 | 0.8914 | 0.2406 | 0.4890 | 0.4267 | 0.6861 |
| | Vanilla SAM (Ours) | 0.5962 | 0.7108 | 0.6672 | 0.8962 | 0.2539 | 0.5446 | 0.5058 | 0.7172 |
| | DPAdapter (Ours) | **0.6008** | **0.7186** | **0.6730** | **0.9012** | **0.3023** | **0.5676** | **0.5254** | **0.7291** |
| GEP | None (Scratch) [52] | 0.4008 | 0.4672 | 0.5892 | 0.8158 | 0.2360 | 0.3239 | 0.4087 | 0.5356 |
| | Standard Pre-training [15, 26] | 0.6512 | 0.7456 | 0.8078 | 0.8500 | 0.3326 | 0.6002 | 0.5972 | 0.7319 |
| | Vanilla SAM (Ours) | 0.6808 | 0.7472 | 0.8132 | 0.8538 | 0.3739 | 0.6168 | 0.6226 | 0.7393 |
| | DPAdapter (Ours) | **0.6890** | **0.7692** | **0.8180** | **0.8686** | **0.4730** | **0.6462** | **0.6600** | **0.7613** |
| AdpAlloc | None (Scratch) [53] | 0.4370 | 0.5166 | 0.6248 | 0.7678 | 0.2923 | 0.3391 | 0.4514 | 0.5412 |
| | Standard Pre-training [15, 26] | 0.4506 | 0.6982 | 0.6604 | 0.8938 | 0.2946 | 0.5104 | 0.4685 | 0.7008 |
| | Vanilla SAM (Ours) | 0.5296 | 0.7372 | 0.7652 | 0.8998 | 0.2933 | 0.5675 | 0.5294 | 0.7348 |
| | DPAdapter (Ours) | **0.5352** | **0.7406** | **0.7862** | **0.9008** | **0.2938** | **0.6111** | **0.5384** | **0.7508** |

rameter robustness can additionally enhance downstream accuracy. For example, when the privacy budget is designated as $\varepsilon = 1$ and the DPML algorithm is configured to AdpAlloc, using a standard pre-trained model improves the average downstream accuracy by 1.71%. Meanwhile, applying vanilla SAM can further augment the average downstream accuracy by 6.09% (compared with standard pre-training), achieving an additional gap that is over three times larger than the previous gap achieved by standard pre-training. This underscores the significance of leveraging parameter robustness to enhance DPML algorithms.

Moreover, we observe that employing DPAdapter consistently achieves higher downstream accuracy compared to using vanilla SAM. For instance, when the privacy budget is set to $\varepsilon = 4$ and the DPML algorithm is configured to GEP, applying vanilla SAM enhances the average downstream accuracy by 0.74% compared with standard pre-training. Meanwhile, applying DPAdapter can further increase the average downstream accuracy by 2.20% (compared with vanilla SAM). When the privacy budget is $\varepsilon = 1$, applying vanilla SAM improves the average downstream accuracy by 2.54%; DPAdapter, in contrast, achieves an improvement of 6.28% compared with standard pre-training. This emphasizes the importance of leveraging decoupled batches to further enhance parameter robustness.

We can also observe that GEP is particularly suitable for DPAdapter, potentially due to the specific challenges posed by DPML, notably the "curse of dimensionality". This phenomenon indicates that as the model dimension increases, the required noise addition also escalates, significantly impacting model performance. GEP mitigates this challenge by reducing dimensionality and applying noise in a lower-dimensional space, effectively diminishing the adverse effects of noise on model performance. In contrast, DPAdapter focuses on enhancing each dimension's resilience to noise, thereby maintaining model performance even as noise increases in larger dimensions. The integration of GEP and DPAdapter appears especially effective for high-dimensional models, leveraging their distinct methodologies. This synergy arises from their complementary approaches to managing noise and dimensionality, showcasing how combining these strategies can address the challenges of pre-trained models more efficiently.

## 5.3 Impact of Perturbation Magnitude

Recall that DPAdapter fundamentally accumulates parameter robustness by introducing adversarial perturbation during the training process, the magnitude of which is determined by the $\gamma$ term. To comprehend how DPAdapter influences the performance of downstream tasks, we assess DPAdapter utilizing various $\gamma$ settings. In this experiment, we fix the privacy budget at $\varepsilon = 1$ and the DPML algorithm is configured to vanilla DP-SGD.

**Observations**. Figure 5 illustrates the fluctuations in upstream performance, upstream parameter robustness, and the performance of various downstream tasks when applying DPAdapter with different $\gamma$ values, representing the perturbation magnitude. We observe that the downstream accuracy

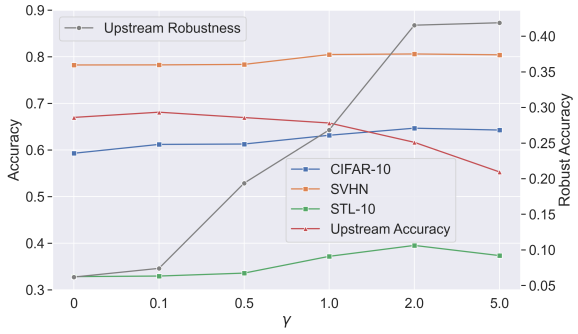


Figure 5: Impact of perturbation magnitude on DPAdapter.

trend largely aligns with the trend of upstream parameter robustness, while demonstrating a nearly inverse relationship with the level of upstream accuracy. This observation suggests that parameter robustness is the principal factor through which DPAdapter enhances downstream accuracy. Given that the trends of upstream accuracy and downstream accuracy exhibit nearly inverse patterns with changes in $\gamma$, we can essentially rule out the possibility that upstream accuracy is the predominant contributor to downstream accuracy.

Note that when $\gamma$ is further increased to 5.0, the upstream accuracy experiences a slight increase while the performance across different downstream tasks all decline. This could be attributed to the scenario where the enhancement in parameter robustness becomes constrained and the decline in upstream accuracy becomes significant. The advantages conferred by improved parameter robustness are unable to offset the disadvantages brought about by the reduction in upstream accuracy, since upstream accuracy also contributes to downstream accuracy through enhanced generalization ability. This underscores the vital necessity of maintaining a balance between parameter robustness and generalization ability.

In this experiment, for the first time, we identify and establish a connection between parameter robustness and the performance of DPML algorithms, marking one of the key contributions of this paper.

## 5.4 Comparison with Vanilla SAM

**Setup**. In this experiment, we consider four perturbation magnitudes: 0.1, 0.5, 1.0, and 2.0. The DPML algorithm is fixed at vanilla DP-SGD and the privacy budget is fixed at $\varepsilon = 1$. The results of using the vanilla SAM are denoted by blue triangles, while the results of the proposed DPAdapter are denoted by red diamonds. The result of using a normally pre-trained model is denoted as a purple rectangle.

**Observations**. Figure 6 illustrate the influence of perturbation magnitude on both vanilla SAM and DPAdapter. The x-axis represents the robust accuracy, which measures parameter robustness, while the y-axis indicates the accuracy, reflecting model performance. We observe that models using standard pre-training typically appear in the bottom-left region, demonstrating a weak robustness and generalization trade-off, identified as crucial for DPML algorithms in Section 5.3. Conversely, the results derived from the proposed DPAdapter predominantly reside in the top-right region, showcasing an enhancement over the vanilla SAM. This implies that DPAdapter can further refine the robustness-generalization trade-off compared to vanilla SAM, which may elucidate its consistent ability to amplify the performance of DPML algorithms with the most substantial improvements.

Table 2: The impact of public dataset and model architecture.

| Public Dataset → | | CIFAR-100 | | Tiny ImageNet | |
|---|---|---|---|---|---|
| Model Arch. ↓ | Pre-training Method ↓ | $\varepsilon = 1$ | $\varepsilon = 4$ | $\varepsilon = 1$ | $\varepsilon = 4$ |
| ResNet20 | Standard | 0.3282 | 0.5695 | 0.3129 | 0.5460 |
| | DPAdapter | **0.3951** | **0.6364** | **0.3671** | **0.6031** |
| ViT-B | Standard | 0.4331 | 0.4929 | 0.5499 | 0.6118 |
| | DPAdapter | **0.4480** | **0.5200** | **0.5730** | **0.6374** |

Table 3: Compatibility of DPAdapter with Downstream Enhancements.

| DPML Algorithm ↓ | Pre-training Method ↓ | $\varepsilon = 1$ | $\varepsilon = 4$ |
|---|---|---|---|
| DP-SGD | Standard | 0.3282 | 0.5695 |
| | DPAdapter | **0.3951** | **0.6364** |
| DP-SGD + MGC [7] | Standard | 0.3397 | 0.5817 |
| | DPAdapter | **0.4109** | **0.6432** |

## 5.5 Impact of Public Dataset and Model Architecture

**Setup**. In this experiment, we explore the effects of public dataset selection and model architecture on DPAdapter. We evaluate two public datasets: Tiny ImageNet and CIFAR-100, alongside two model architectures: ResNet20 and ViT-B. The downstream task is fixed as STL-10. For the fine-tuning of ViT-B, all layers except the linear ones are frozen. The outer learning rate, $\eta_2$, is set to 0.05 for experiments involving Tiny ImageNet.

**Observations**. The results, as shown in Table 2, reveal the influence of public dataset selection and model architecture on the effectiveness of DPAdapter. Overall, DPAdapter enhances test accuracy across all configurations, showcasing its adaptability. Notably, Tiny ImageNet appears to be more compatible with ViT-B, whereas CIFAR-100 is better suited for ResNet20. This discrepancy may be attributed to the resolution differences between the datasets, given that ViT-B segments an image into 16x16 tiles.

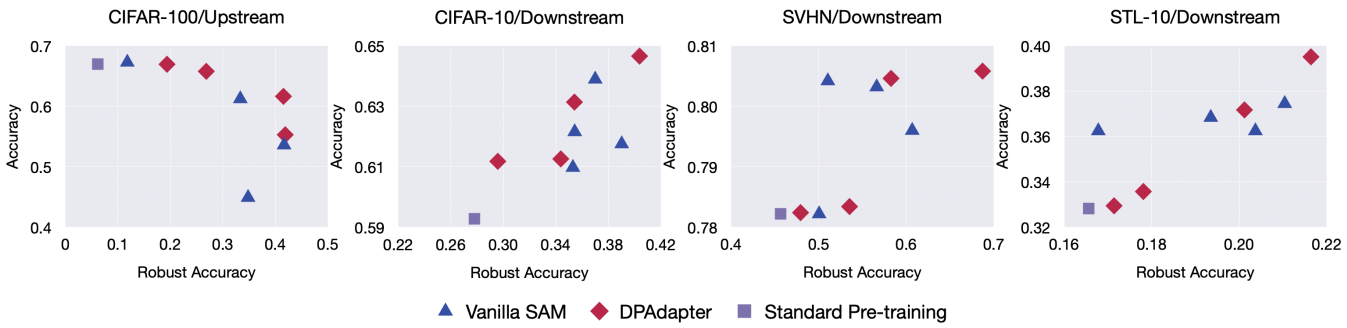


Figure 6: The results of DPAdapter and vanilla SAM under different perturbation magnitudes.

### 5.6 Compatibility with Other Enhancements

**Setup**. In this experiment, we investigate the compatibility of DPAdapter with downstream enhancements, focusing on the state-of-the-art DP-SGD enhancement method, mixed ghost clipping (MGC), as proposed in recent research [7]. We designate STL-10 as the downstream task and CIFAR-100 as the public dataset for this evaluation.

**Observations**. The results, as presented in Table 3, reveal that both DPAdapter and MGC effectively improve test accuracy. When combined, they achieve even better test accuracy, indicating that DPAdapter is highly compatible with downstream enhancements. Furthermore, it is noted that DPAdapter consistently offers a more significant enhancement compared to MGC. Specifically, the accuracy improvement attributed to MGC is around 1% for both $\varepsilon = 1$ and $\varepsilon = 4$ scenarios. In contrast, DPAdapter consistently yields an enhancement of over 6% across all cases, demonstrating its superior efficacy in enhancing model performance.

## 6 Discussion

The current design of DPAdapter is specifically targeted towards supervised learning and cannot be directly applied to unsupervised learning techniques like contrastive learning [11, 12, 21]. Thus, the attacker in our scenario needs to have a labeled dataset for pre-training, which might pose a challenge in certain specialized fields.

Moreover, while our discussion and implementation center on a single-party scenario, it is worth noting that our framework is adaptable to both single-party and multi-party situations. In a multi-party context, the private data originates from multiple sources. Consequently, private fine-tuning processes can be facilitated through federated learning [28].

## 7 Related Work

**Parameter Robustness**. Numerous studies have shown that enhancing parameter robustness effectively narrows the generalization gap, leading to improved model generalizability [17, 29, 55]. In this paper, we categorize these efforts under the umbrella of sharpness-aware minimization (SAM) approaches. However, all of these studies utilize the additional benefits conferred by parameter robustness, i.e., generality, without directly leveraging the property of parameter robustness itself. In contrast, the proposed DPAdapter is among the first strategies that directly utilize parameter robustness to benefit private learning. This introduces a fresh perspective on the potential applications of the parameter robustness property. Several pieces of research [41, 45] have incorporated SAM into DP algorithms to improve test accuracy. Yet, their protection is applied during the fine-tuning phase, whereas DPAdapter implements protection in the pre-training phase. It is important to recognize that each approach brings its own advantages, and combining DPAdapter with these methods could potentially boost accuracy further. Therefore, the protection offered at the pre-training stage deserves special attention for its unique contributions.

**Pre-trained Model as a Service**. Adversarial training [19, 33] is recognized as a standard method for developing empirically robust classifiers in supervised learning. The core concept involves generating adversarial examples from training instances during the training process and augmenting the training data with these examples. Several studies [10, 23, 24, 43] have generalized adversarial training for the pre-training of robust models in self-supervised learning. Generally, the approach first generates adversarial examples that result in significant loss, which are then used for model pre-training. However, previous studies have focused solely on offering input robustness as a service, neglecting to offer parameter robustness, which is the primary contribution of this paper.

## 8 Conclusion

In this study, we unveiled DPAdapter, a groundbreaking technique engineered to augment parameter robustness, thereby navigating the traditionally adversarial relationship between Differential Privacy (DP) noise and model utility in Deep Learning. By meticulously reallocating batch sizes for

perturbation and gradient calculations, DPAdapter refines Sharpness-Aware Minimization (SAM) algorithms, delivering enhanced parameter robustness and, consequently, mitigating the impact of DP noise. Our comprehensive evaluations across several datasets substantiate DPAdapter's capability to substantially bolster the accuracy of DPML algorithms on various downstream tasks, thereby highlighting its potential as a pivotal technique in future privacy-preserving machine learning endeavors.

## Acknowledgements

We sincerely thank our shepherd and the anonymous reviewers for their valuable feedback. Authors from Indiana University were supported in part by IARPA W91NF-20-C-0034 (the TrojAI project).

## 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 *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 308–318, 2016.

[2] G. Andrew, O. Thakkar, B. McMahan, and S. Ramaswamy. Differentially private learning with adaptive clipping. *Advances in Neural Information Processing Systems*, 34:17455–17466, 2021.

[3] M. Andriushchenko and N. Flammarion. Towards understanding sharpness-aware minimization. In *International Conference on Machine Learning*, pages 639–668. PMLR, 2022.

[4] A. Athalye, N. Carlini, and D. A. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 274–283, 2018.

[5] B. Balle, G. Cherubin, and J. Hayes. Reconstructing training data with informed adversaries. In *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022*, pages 1138–1156, 2022.

[6] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Srndic, P. Laskov, G. Giacinto, and F. Roli. Evasion attacks against machine learning at test time. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III*, pages 387–402, 2013.

[7] Z. Bu, J. Mao, and S. Xu. Scalable and efficient training of large convolutional neural networks with differential privacy. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[8] N. Carlini, S. Chien, M. Nasr, S. Song, A. Terzis, and F. Tramèr. Membership inference attacks from first principles. In *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022*, pages 1897–1914, 2022.

[9] N. Carlini, F. Tramèr, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. B. Brown, D. Song, Ú. Erlingsson, A. Oprea, and C. Raffel. Extracting training data from large language models. In *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, pages 2633–2650, 2021.

[10] T. Chen, S. Liu, S. Chang, Y. Cheng, L. Amini, and Z. Wang. Adversarial robustness: From self-supervised pre-training to fine-tuning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 696–705, 2020.

[11] X. Chen, H. Fan, R. B. Girshick, and K. He. Improved baselines with momentum contrastive learning. *CoRR*, abs/2003.04297, 2020.

[12] X. Chen, S. Xie, and K. He. An empirical study of training self-supervised vision transformers. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 9620–9629, 2021.

[13] A. Coates, A. Y. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, pages 215–223, 2011.

[14] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. P. Kuksa. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, 2011.

[15] S. De, L. Berrada, J. Hayes, S. L. Smith, and B. Balle. Unlocking high-accuracy differentially private image classification through scale. *CoRR*, abs/2204.13650, 2022.

[16] C. Dwork, F. McSherry, K. Nissim, and A. D. Smith. Calibrating noise to sensitivity in private data analysis.

In *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, pages 265–284, 2006.

[17] P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

[18] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[19] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[20] P. Goyal, P. Dollár, R. B. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. Accurate, large minibatch SGD: training imagenet in 1 hour. *CoRR*, abs/1706.02677, 2017.

[21] K. He, H. Fan, Y. Wu, S. Xie, and R. B. Girshick. Momentum contrast for unsupervised visual representation learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 9726–9735, 2020.

[22] 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, 2016.

[23] Z. Jiang, T. Chen, T. Chen, and Z. Wang. Robust pre-training by adversarial contrastive learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[24] M. Kim, J. Tack, and S. J. Hwang. Adversarial self-supervised contrastive learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[25] A. Krizhevsky. Learning multiple layers of features from tiny images. Tech. report, University of Toronto, 2009.

[26] A. Kurakin, S. Chien, S. Song, R. Geambasu, A. Terzis, and A. Thakurta. Toward training at imagenet scale with differential privacy. *CoRR*, abs/2201.12328, 2022.

[27] F.-F. Li, A. Karpathy, and J. Johnson. Cs231n: Convolutional neural networks for visual recognition. Course at Stanford University, 2016.

[28] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Process. Mag.*, 37(3):50–60, 2020.

[29] T. Li, W. Yan, Z. Lei, Y. Wu, K. Fang, M. Yang, and X. Huang. Efficient generalization improvement guided by random weight perturbation. *CoRR*, abs/2211.11489, 2022.

[30] X. Li, F. Tramèr, P. Liang, and T. Hashimoto. Large language models can be strong differentially private learners. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022.

[31] Z. Li and Y. Zhang. Membership leakage in label-only exposures. In *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, pages 880–895, 2021.

[32] Y. Liu, R. Wen, X. He, A. Salem, Z. Zhang, M. Backes, E. D. Cristofaro, M. Fritz, and Y. Zhang. Ml-doctor: Holistic risk assessment of inference attacks against machine learning models. In *31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*, pages 4525–4542, 2022.

[33] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[34] H. Mehta, A. Thakurta, A. Kurakin, and A. Cutkosky. Large scale transfer learning for differentially private image classification. *CoRR*, abs/2205.02973, 2022.

[35] I. Mironov, K. Talwar, and L. Zhang. Rényi differential privacy of the sampled gaussian mechanism. *CoRR*, abs/1908.10530, 2019.

[36] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2843–2851, 2011.

[37] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.*, 22(10):1345–1359, 2010.

[38] N. Papernot, M. Abadi, Ú. Erlingsson, I. J. Goodfellow, and K. Talwar. Semi-supervised knowledge transfer for deep learning from private training data. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[39] N. Papernot, M. Abadi, Ú. Erlingsson, I. J. Goodfellow, and K. Talwar. Semi-supervised knowledge transfer for deep learning from private training data. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[40] N. Papernot, P. D. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. The limitations of deep learning in adversarial settings. In *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016*, pages 372–387, 2016.

[41] J. Park, H. Kim, Y. Choi, and J. Lee. Differentially private sharpness-aware training. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, pages 27204–27224, 2023.

[42] W. Qu, J. Jia, and N. Z. Gong. Reaas: Enabling adversarially robust downstream classifiers via robust encoder as a service. *arXiv preprint arXiv:2301.02905*, 2023.

[43] W. Qu, J. Jia, and N. Z. Gong. Reaas: Enabling adversarially robust downstream classifiers via robust encoder as a service. In *30th Annual Network and Distributed System Security Symposium, NDSS 2023, San Diego, California, USA, February 27 - March 3, 2023*, 2023.

[44] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 91–99, 2015.

[45] Y. Shi, Y. Liu, K. Wei, L. Shen, X. Wang, and D. Tao. Make landscape flatter in differentially private federated learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 24552–24562, 2023.

[46] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[47] F. Tramèr and D. Boneh. Differentially private learning needs better features (or much more data). In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

[48] C. Wei, M. Zhao, Z. Zhang, M. Chen, W. Meng, B. Liu, Y. Fan, and W. Chen. Dpmlbench: Holistic evaluation of differentially private machine learning. *CoRR*, abs/2305.05900, 2023.

[49] D. Wu, S. Xia, and Y. Wang. Adversarial weight perturbation helps robust generalization. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[50] D. Yu, S. Naik, A. Backurs, S. Gopi, H. A. Inan, G. Kamath, J. Kulkarni, Y. T. Lee, A. Manoel, L. Wutschitz, S. Yekhanin, and H. Zhang. Differentially private fine-tuning of language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022.

[51] D. Yu, H. Zhang, W. Chen, and T. Liu. Do not let privacy overbill utility: Gradient embedding perturbation for private learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

[52] D. Yu, H. Zhang, W. Chen, and T.-Y. Liu. Do not let privacy overbill utility: Gradient embedding perturbation for private learning. In *International Conference on Learning Representations (ICLR)*, 2021.

[53] 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)*, pages 332–349. IEEE, 2019.

[54] J. Zhang, K. Zheng, W. Mou, and L. Wang. Efficient private erm for smooth objectives. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 3922–3928, 2017.

[55] Y. Zheng, R. Zhang, and Y. Mao. Regularizing neural networks via adversarial model perturbation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 8156–8165, 2021.

[56] Y. Zhu, X. Yu, M. Chandraker, and Y. Wang. Private-knn: Practical differential privacy for computer vision. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 11851–11859, 2020.

# Appendix

## A Theoretical Results

### A.1 Proof of Theorem 1

We first restate Theorem Theorem 1 as follows.

**Theorem 3** (Formal version of Theorem 1)**.** *Denote* $\hat{\beta} := (\rho^2\beta_2 + \beta\beta_1)$. *Select step sizes* $\eta_1 = \eta_2 = 1/(4\hat{\beta})$. *Then Algorithm 1 enjoys the following utility bound:*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}(L_{\mathcal{D}}(\boldsymbol{\theta}_t) - \inf_{\hat{\boldsymbol{\theta}}} L_{\mathcal{D}}(\hat{\boldsymbol{\theta}})) \leq \mu \cdot \left(\frac{L_{\mathcal{D}}(\boldsymbol{\theta}_0)}{T} + \frac{\hat{\sigma}^2}{16\hat{\beta}}(1/|\mathcal{B}_2| + 1/|\mathcal{B}_1|)\right).$$

To prove Theorem 3, we need the following lemma which estimates the Lipschitz constant and smoothness constant of $L_i$.

**Lemma A.1.** *For each i,* $L_i$ *is* $\beta_1\rho$*-Lipschitz continuous and* $(\rho^2\beta_2 + \beta\beta_1)$*-smooth.*

*Proof.* We have for all $\boldsymbol{\theta}, \boldsymbol{\theta}'$,

$$\begin{aligned}
|L_i(\boldsymbol{\theta}) - L_i(\boldsymbol{\theta}')| &= |\ell(f_{\boldsymbol{\theta}}(x_i), y_i) - \ell(f_{\boldsymbol{\theta}'}(x_i), y_i)| \\
&\leq \beta_1 |f_{\boldsymbol{\theta}}(x_i) - f_{\boldsymbol{\theta}'}(x_i)| \\
&\leq \beta_1\rho\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2,
\end{aligned}$$

and

$$\begin{aligned}
&\|\nabla L_i(\boldsymbol{\theta}) - \nabla L_i(\boldsymbol{\theta}')\|_2 \\
&= \|\nabla_x \ell(f_{\boldsymbol{\theta}}(x_i), y_i)\nabla f_{\boldsymbol{\theta}}(x_i) - \nabla_x \ell(f_{\boldsymbol{\theta}'}(x_i), y_i)\nabla f_{\boldsymbol{\theta}'}(x_i)\|_2 \\
&\leq \|\nabla_x \ell(f_{\boldsymbol{\theta}}(x_i), y_i)\nabla f_{\boldsymbol{\theta}}(x_i) - \nabla_x \ell(f_{\boldsymbol{\theta}'}(x_i), y_i)\nabla f_{\boldsymbol{\theta}}(x_i)\|_2 + \|\nabla_x \ell(f_{\boldsymbol{\theta}'}(x_i), y_i)\nabla f_{\boldsymbol{\theta}}(x_i) - \nabla_x \ell(f_{\boldsymbol{\theta}'}(x_i), y_i)\nabla f_{\boldsymbol{\theta}'}(x_i)\|_2 \\
&\leq \|\nabla f_{\boldsymbol{\theta}}(x_i)\|_2 |\nabla_x \ell(f_{\boldsymbol{\theta}}(x_i), y_i) - \nabla_x \ell(f_{\boldsymbol{\theta}'}(x_i), y_i)| + |\nabla_x \ell(f_{\boldsymbol{\theta}'}(x_i), y_i)| \|\nabla f_{\boldsymbol{\theta}}(x_i) - \nabla f_{\boldsymbol{\theta}'}(x_i)\|_2 \\
&\leq \rho^2\beta_2\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2 + \beta_1\beta\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2.
\end{aligned}$$

Therefore, $L_i$ is $\beta_1\rho$-Lipschitz continuous and $(\rho^2\beta_2 + \beta\beta_1)$-smooth. □

We now begin to prove Theorem 3.

*Proof of Theorem 3.* The proof is adapted from [3]. First, by Lemma A.1 we know that $L_i$ is $\beta_1\rho$-Lipschitz continuous and $(\rho^2\beta_2 + \beta\beta_1)$-smooth. By Lemma 13 in [3], we have

$$\mathbb{E}\langle \nabla L_{\mathcal{B}_2}(\boldsymbol{\theta}_t + \eta_2 \cdot \nabla L_{\mathcal{B}_1}(\boldsymbol{\theta}_t)), \nabla L_{\mathcal{D}}(\boldsymbol{\theta}_t)\rangle \geq (1 - \hat{\beta}\eta_2)\|L_{\mathcal{D}}(\boldsymbol{\theta}_t)\|_2^2 - \frac{\hat{\beta}^2\eta_2^2\hat{\sigma}^2}{2|\mathcal{B}_1|}. \tag{7}$$

By Lemma 14 in [3], denote $\boldsymbol{\theta}_{t+1/2} := \boldsymbol{\theta}_t + \eta_2 \cdot \nabla L_{\mathcal{B}_1}(\boldsymbol{\theta}_t)$, then we have for all $\eta_1 \leq 1/(2\hat{\beta})$ and $\eta_2 \leq 1/(2\hat{\beta})$,

$$\begin{aligned}
\mathbb{E}L_{\mathcal{D}}(\boldsymbol{\theta}_{t+1}) &\leq \mathbb{E}L_{\mathcal{D}}(\boldsymbol{\theta}_t) + \eta_1^2\hat{\beta}\hat{\sigma}^2/|\mathcal{B}_2| - \eta_1^2\hat{\beta}\mathbb{E}\|\nabla L_{\mathcal{D}}(\boldsymbol{\theta}_t)\|_2^2 - \eta_1(1 - 2\eta_1\hat{\beta})\mathbb{E}\langle \nabla L_{\mathcal{D}}(\boldsymbol{\theta}_{t+1/2}), \nabla L_{\mathcal{D}}(\boldsymbol{\theta}_t)\rangle \\
&\quad + \eta_1^2\hat{\beta}\mathbb{E}\|\nabla L_{\mathcal{D}}(\boldsymbol{\theta}_{t+1/2}) - \nabla L_{\mathcal{D}}(\boldsymbol{\theta}_t)\|_2^2 \\
&\leq \mathbb{E}L_{\mathcal{D}}(\boldsymbol{\theta}_t) + \eta_1^2\hat{\beta}\hat{\sigma}^2/|\mathcal{B}_2| - \eta_1^2\hat{\beta}\mathbb{E}\|\nabla L_{\mathcal{D}}(\boldsymbol{\theta}_t)\|_2^2 - \eta_1(1 - 2\eta_1\hat{\beta})\mathbb{E}\langle \nabla L_{\mathcal{D}}(\boldsymbol{\theta}_{t+1/2}), \nabla L_{\mathcal{D}}(\boldsymbol{\theta}_t)\rangle \\
&\quad + \eta_1^2\hat{\beta}\mathbb{E}\|\boldsymbol{\theta}_{t+1/2} - \boldsymbol{\theta}_t\|_2^2 \\
&= \mathbb{E}L_{\mathcal{D}}(\boldsymbol{\theta}_t) + \eta_1^2\hat{\beta}\hat{\sigma}^2/|\mathcal{B}_2| - \eta_1^2\hat{\beta}\mathbb{E}\|\nabla L_{\mathcal{D}}(\boldsymbol{\theta}_t)\|_2^2 - \eta_1(1 - 2\eta_1\hat{\beta})\mathbb{E}\langle \nabla L_{\mathcal{D}}(\boldsymbol{\theta}_{t+1/2}), \nabla L_{\mathcal{D}}(\boldsymbol{\theta}_t)\rangle \\
&\quad + \eta_1^2\hat{\beta}^3\eta_2^2\mathbb{E}\|\nabla L_{\mathcal{B}_1}(\boldsymbol{\theta}_t)\|_2^2 \\
&\leq \mathbb{E}L_{\mathcal{D}}(\boldsymbol{\theta}_t) + \eta_1^2\hat{\beta}\hat{\sigma}^2/|\mathcal{B}_2| - \eta_1^2\hat{\beta}(1 - 2\hat{\beta}^2\eta_2^2)\mathbb{E}\|\nabla L_{\mathcal{D}}(\boldsymbol{\theta}_t)\|_2^2 - \eta_1(1 - 2\eta_1\hat{\beta})\mathbb{E}\langle \nabla L_{\mathcal{D}}(\boldsymbol{\theta}_{t+1/2}), \nabla L_{\mathcal{D}}(\boldsymbol{\theta}_t)\rangle \\
&\quad + 2\eta_1^2\hat{\beta}^3\eta_2^2\hat{\sigma}^2/|\mathcal{B}_1|.
\end{aligned}$$

**Algorithm 2** Random round DP-SGD

**Require:** Training set $\mathcal{D} = \{(\boldsymbol{x}, \boldsymbol{y})\}$, loss function $L_i := \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i)$, privacy parameter $(\varepsilon, \delta)$, learning rate $\eta$.

1: Randomly draw $R \sim \mathbb{P}$, where

$$\mathbb{P}(R = k+1) := \frac{1}{|\mathcal{D}|^2}, k = 0, 1, \ldots, |\mathcal{D}|^2 - 1.$$

2: **for** $t = 0, \ldots, R-1$ **do**

3: Uniformly randomly draw $i$ from $1, \ldots, |\mathcal{D}|$

4: Draw $z_t \in \mathbb{R}^d$ from the following Gaussian distribution:

$$z^t \sim \mathcal{N}(0, 4\beta_1^2 \rho^2 \log(3|\mathcal{D}|/\delta) \log(2/\delta) \cdot \mathbf{I}) \tag{9}$$

5: Update $\boldsymbol{\theta}_{t+1} := \boldsymbol{\theta}_t - \eta \cdot (\nabla L_i(\boldsymbol{\theta}_t) + z_t)$

6: **end for**

**Ensure:** $\boldsymbol{\theta}_{\text{out}} = \boldsymbol{\theta}_R$

Then substituting Equation 7 and setting $\eta_1 = \eta_2 = 1/(4\hat{\beta})$, we have

$$\begin{aligned}
&\mathbb{E}L_{\mathcal{D}}(\boldsymbol{\theta}_{t+1}) - \mathbb{E}L_{\mathcal{D}}(\boldsymbol{\theta}_t) \\
&\leq \hat{\sigma}^2/(16|\mathcal{B}_2|\hat{\beta}) - 1/(16\hat{\beta}) \cdot 7/8 \cdot \mathbb{E}\|\nabla L_{\mathcal{D}}(\boldsymbol{\theta}_t)\|_2^2 - 1/(8\hat{\beta}) \cdot (3/4 \cdot \mathbb{E}\|\nabla L_{\mathcal{D}}(\boldsymbol{\theta}_t)\|_2^2 - \hat{\sigma}^2/(32|\mathcal{B}_1|)) + \hat{\sigma}^2/(128\hat{\beta}|\mathcal{B}_1|) \\
&\leq \hat{\sigma}^2/(64\hat{\beta}|\mathcal{B}_1|) + \hat{\sigma}^2/(16\hat{\beta}|\mathcal{B}_2|) - 1/(8\hat{\beta}) \cdot \mathbb{E}\|\nabla L_{\mathcal{D}}(\boldsymbol{\theta}_t)\|_2^2.
\end{aligned}$$

Therefore, we have

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\|\nabla L_{\mathcal{D}}(\boldsymbol{\theta}_t)\|_2^2 \leq \frac{L_{\mathcal{D}}(\boldsymbol{\theta}_0)}{T} + \frac{\hat{\sigma}^2}{16\hat{\beta}}(1/|\mathcal{B}_2| + 1/|\mathcal{B}_1|). \tag{8}$$

Finally, by using the PL-condition we have $\|\nabla L_{\mathcal{D}}(\boldsymbol{\theta}_t)\|_2^2 \geq 1/\mu \cdot (L_{\mathcal{D}}(\boldsymbol{\theta}_t) - \inf_{\boldsymbol{\theta}} L_{\mathcal{D}}(\boldsymbol{\theta}))$. Substituting it into (8) concludes the proof. □

## A.2 Proof of Theorem 2

We present the random round DP-SGD proposed in [54] as Algorithm 2. Compared with the standard DP-SGD [1], the main difference is the random selected round number $R$ here. The use of a random round number is only due to the need of theoretical proof. Next, we state the formal version of Theorem Theorem 2 as follows.

**Theorem 4** (Formal version of Theorem 2)**.** *Select the step size* $\eta = \min\{1/(\rho^2\beta_2 + \beta\beta_1), D_f/(\sigma|\mathcal{D}|)\}$*, where*

$$D_f := \sqrt{2(L_{\mathcal{D}}(\boldsymbol{\theta}_0) - \min_{\boldsymbol{\theta}} L_{\mathcal{D}}(\boldsymbol{\theta}))/(\rho^2\beta_2 + \beta\beta_1)},$$
$$\sigma := 2\beta_1\rho\sqrt{1 + \frac{d\log(3|\mathcal{D}|/\delta)\log(2\delta)}{\varepsilon^2}},$$

*then we have the following utility bound:*

$$\mathbb{E}(L_{\mathcal{D}}(\boldsymbol{\theta}_{out}) - \inf_{\boldsymbol{\theta}} L_{\mathcal{D}}(\boldsymbol{\theta})) \leq C \cdot \frac{\rho\sqrt{\rho^2\beta_2 + \beta\beta_1}}{|\mathcal{D}|\varepsilon},$$

*where* $C = c \cdot \mu\beta_1\sqrt{d\log(|\mathcal{D}|/\delta)\log(1/\delta)L_{\mathcal{D}}(\boldsymbol{\theta}_0)}$*, c is some positive constant, d is the dimension of parameter* $\boldsymbol{\theta}$*.*

*Proof of Theorem 4.* First, by Lemma A.1, we have $L_i$ is $\beta_1\rho$-Lipschitz continuous and $(\rho^2\beta_2 + \beta\beta_1)$-smooth. Then according to Theorem 5 in [54], we have the following utility bound:

$$\mathbb{E}\|\nabla L_{\mathcal{D}}(\boldsymbol{\theta}_{\text{out}})\|_2^2 \leq c \cdot \left(\frac{\beta_1\rho\sqrt{\rho^2\beta_2 + \beta\beta_1}\sqrt{d\log(|\mathcal{D}|/\delta)\log(1/\delta)L_{\mathcal{D}}(\boldsymbol{\theta}_0)}}{|\mathcal{D}|\varepsilon}\right), \tag{10}$$

where $c$ is some positive constant. Meanwhile, random round DP-SGD is $(\varepsilon, \delta)$-DP due to Theorem 4 in [54]. Finally, by using the PL-condition we have $\|\nabla L_{\mathcal{D}}(\boldsymbol{\theta}_{\text{out}})\|_2^2 \geq 1/\mu \cdot (L_{\mathcal{D}}(\boldsymbol{\theta}_{\text{out}}) - \inf_{\boldsymbol{\theta}} L_{\mathcal{D}}(\boldsymbol{\theta}))$. Substituting it into Equation 10 concludes the proof. □