DP-FROST: Differentially Private Fine-tuning of Pre-trained Models with Freezing Model Parameters

Daeyoung Hong
Myongji University
South Korea
dyhong@mju.ac.kr

Woohwan Jung
Hanyang University
South Korea
whjung@hanyang.ac.kr

Kyuseok Shim*
Seoul National University
South Korea
kshim@snu.ac.kr

Abstract

Training models with differential privacy has received a lot of attentions since differential privacy provides theoretical guarantee of privacy preservation. For a task in a specific domain, since a large-scale pre-trained model in the same domain contains general knowledge of the task, using such a model requires less effort in designing and training the model. However, differentially privately fine-tuning such models having a large number of trainable parameters results in large degradation of utility. Thus, we propose methods that effectively fine-tune the large-scale pre-trained models with freezing unimportant parameters for downstream tasks while satisfying differential privacy. To select the parameters to be fine-tuned, we propose several efficient methods based on the gradients of model parameters. We show the effectiveness of the proposed method by performing experiments with real datasets[1].

1 Introduction

Deep learning models conventionally trained on private user data have risks of leaking sensitive information (Shokri et al., 2017; Carlini et al., 2019, 2021). To protect sensitive statistical data, the differential privacy (Dwork and Roth, 2013) is widely used due to theoretical guarantee of privacy preservation by limiting the influence of any individual user's data and injecting noise into the results computed from the sensitive data. The differential privacy has also attracted attentions in the machine learning community to build a differentially private model (Abadi et al., 2016).

To satisfy differential privacy, deep learning models are generally trained by the DP-SGD algorithm (Abadi et al., 2016). Since the amount of inserted noise into the gradients grows with increasing the number of model parameters (Luo et al., 2021), fine-tuning large pre-trained models may significantly degrade their accuracies. Thus, LoRA (Li et al., 2022) adds a small-sized low-rank matrix to each existing pre-trained weight matrix in a model and fine-tunes only the low-rank matrices. Similarly, Adapter (Yu et al., 2022) adds a small-sized intermediate layer after each existing layer of the pre-trained model, and fine-tunes only the intermediate layers. However, both methods fine-tune only added parameters without considering the impact of parameters to improve the accuracy. On the other hand, the method in (Luo et al., 2021) freezes the pre-trained model parameters with small absolute values and fine-tune only the other parameters. But updating such freezed parameters may significantly improve the accuracy of the trained model.

To overcome the drawbacks of existing works, we propose the DP-FROST (Differentially Private Fine-tuning of pRe-trained mOdelS with freezing model parameTers) method. The proposed method (1) splits the model parameters into disjoint partitions, (2) adds noise to the partition-wise gradient magnitude, (3) selects partitions with their largest noisy partition-wise gradient magnitudes and (4) updates the parameters in the selected partitions.

Our contributions: The contributions of this paper are summarized below.

- We prove that the model parameters with large gradient magnitudes contribute more to its accuracy than those with small gradient magnitudes.
- To reduce the amount of inserted noise, instead of selecting the parameters in the granularity of parameters by adding noise to their gradient magnitudes, we propose to select the parameters in the granularity of partitions by adding noise only to the partition-wise gradient magnitude.
- We propose to compute the noisy partition-wise gradient magnitude (PGM) by (1) taking absolute value in every dimension of per-example gradients, (2) normalizing and clipping the re-

*Corresponding author

[1]Our codes will be available at https://github.com/daeyounghong/dp-frost

Proceedings of the 31st International Conference on Computational Linguistics, pages 6966–6984
January 19–24, 2025.

sulting per-example gradients and (3) adding noise to the partition-wise norm of the sum vector of the resulting per-example gradients. We prove that normalizing before clipping reduces the variance of inserted noise.

- To reduce the effect of added noise further, we propose to iteratively select unfreezed partitions and compute noisy PGMs repeatedly based on remaining partitions only in each selection. We formulate the problem of estimating the true PGMs with the observed noisy PGMs computed for each partition as a maximum likelihood estimation and use the solution of the optimization problem to select unfreezed partitions.
- Experiments with real datasets show that our method outperforms the existing methods in terms of the accuracy of the fine-tuned models.

2 Related Works

Since ϵ-differential privacy (Dwork et al., 2006) was introduced to protect sensitive statistical data from queries by users, it has been extensively used to collect or publish various statistical data such as geospatial data (Qardaji et al., 2013) and medical data (Sun et al., 2019). On the other hand, the machine learning community utilizes (ϵ, δ)-differential privacy (Dwork and Roth, 2013), which is a relaxed variant of the ϵ-differential privacy. For instance, deep learning models are trained to satisfy (ϵ, δ)-differential privacy by utilizing the DP-SGD algorithm (Abadi et al., 2016).

When using the DP-SGD algorithm for deep learning, to reduce the noise error, the idea of freezing a subset of model parameters in training is investigated in (Luo et al., 2021; Li et al., 2022; Yu et al., 2022). For example, when fine-tuning a pre-trained deep convolutional neural model, the work in (Luo et al., 2021) freezes the model parameters with small absolute values and updates the unfrozen parameters only by using the DP-SGD algorithm. However, updating the parameters with small absolute values can lead to significant improvement of the accuracy of a trained model. On the other hand, for large pre-trained language models, LoRA proposed in (Li et al., 2022; Yu et al., 2022) adds low-rank matrices to the existing weight matrices in a model and fine-tunes the newly added low-rank matrices. Similarly, Adapter (Yu et al., 2022) adds intermediate layers after each attention and feed-forward layer, and fine-tunes only the intermediate layers. However, it can be more difficult to fine-tune the newly added layers in LoRA or Adapter since they are randomly initialized compared to the pre-trained layers. Moreover, the above three methods select the parameters to fine-tune without utilize the gradients of parameters to estimate their impact on improving accuracy. In contrast, our DP-FROST algorithm select the unfreezed parameters based on the gradient magnitudes of the parameters.

In the federated learning setting, the idea of updating only the parameters with large gradients is considered in (Shokri and Shmatikov, 2015). To satisfy differential privacy while selecting the parameters, it injects noise to the gradients of individual parameters. Since the magnitude of inserted noise grows as the model size increases, for a large model, we may select the parameters with large noisy gradient magnitudes which can be quite different from their original gradient magnitudes.

3 Preliminaries

We introduce (ϵ, δ)-differential privacy and the DP-SGD algorithm that trains a neural model.

3.1 Differential Privacy

We introduce (ϵ, δ)-differential privacy (Dwork and Roth, 2013) and sensitivity of a function.

Definition 1. *A randomized algorithm $\mathcal{A} : \mathcal{D} \to \mathcal{Y}$ satisfies (ϵ, δ)-differential privacy if for any pair of neighboring datasets $D, D' \in \mathcal{D}$, which differ in one individual's data, and for any $Y \subseteq \mathcal{Y}$, we have*

$$\Pr[\mathcal{A}(D) \in Y] \leq e^{\epsilon} \cdot \Pr[\mathcal{A}(D') \in Y] + \delta.$$

For a dataset D and a function f, adding Gaussian noise to the output of the function f achieves (ϵ, δ)-differential privacy and its noise standard deviation is proportional to the sensitivity of f defined below (Mironov, 2017).

Definition 2. *For any two neighboring datasets $D, D' \in \mathcal{D}$, which differ in a single individual, the sensitivity of a function $f : \mathcal{D} \to R^k$, denoted by Δf, is $\Delta f = \max_{D,D'} \|f(D) - f(D')\|_2$.*

3.2 The DP-SGD Algorithm

The DP-SGD algorithm (Abadi et al., 2016) inserts Gaussian noise to the sum of per-example gradients in each mini-batch. Since the standard deviation of noise added to the output of a function is proportional to the sensitivity of the function (Mironov,

2017), instead of directly using the gradient of each example, it uses clipped per-example gradients by scaling the gradient so that their 2-norm do not exceed a clipping threshold c. We provide the definition of the α-norm clipping of a vector v.

Definition 3. *The α-norm clipping of a vector v with a clipping threshold c, denoted by $C_c^\alpha(v)$, is $C_c^\alpha(v) = \min(1, c/\|v\|_\alpha) \cdot v$.*

4 The Proposed Methods

We want to protect the presence of each example in training data based on (ϵ, δ)-differential privacy.

The threat model: Following (Zhang et al., 2020), we assume that the adversary can access the published model including its structure and parameters, but cannot access training processes and train data.

4.1 Overview

The DP-FROST method (1) splits the parameters into disjoint partitions, (2) adds noise to the noisy partition-wise gradient magnitudes (PGMs), (3) selects partitions with their largest noisy PGMs and (4) updates the parameters in the selected partitions.

Parameter partitioning: Instead of inserting noise to the gradient magnitudes of individual parameters, by splitting parameters into partitions and adding noise only to the PGMs, inserted noise to select unfreezed parameters can be reduced.

Computing noisy PGMs: We propose to compute the noisy partition-wise gradient magnitude (PGM) by (1) taking absolute value in every dimension of per-example gradients, (2) normalizing and clipping the resulting per-example gradients and (3) adding noise to the partition-wise norm of the sum vector of the resulting per-example gradients. We prove that normalizing before clipping reduces the variance of inserted noise.

Iterative partition selection: To reduce the effect of added noise further, we propose to iteratively select unfreezed partitions. Since we compute noisy PGMs repeatedly based on remaining partitions only in each selection, we formulate the problem of estimating the true PGMs with the observed noisy PGMs computed for each partition as a maximum likelihood estimation and solve the problem.

Fine-tuning with selected partitions: After selecting the partitions with the largest noisy PGMs, we fine-tune only the parameters in the selected partitions by the DP-SGD algorithm.

4.2 Definitions

We provide the definitions to describe our methods.

Definition 4. *For a parameter set Θ, let $\mathbf{g} \in \mathbb{R}^{|\Theta|}$ be a vector whose i-th element has the value corresponding to the i-th parameter with $1 \leq i \leq |\Theta|$. Then, $[\mathbf{g}]_P$ is the projection of $\mathbf{g}$ onto P s.t. $P \subset \Theta$.*

Definition 5. *For a parameter set $P = \{p_1, p_2, \ldots, p_{|P|}\}$, the gradient of the loss $\mathcal{L}(x)$ of an example x w.r.t. P, denoted by $\mathbf{g}_P(x)$, is $\langle g_{p_1}(x), \cdots, g_{p_{|P|}}(x)\rangle$ with $g_{p_i}(x) = \partial\mathcal{L}(x)/\partial p_i$.*

For an example x, the α-norms $\|\mathbf{g}_\Theta(x)\|_\alpha$ and $\|\mathbf{g}_P(x)\|_\alpha$ of its gradients w.r.t. Θ and $P \subset \Theta$ satisfy $\|\mathbf{g}_\Theta(x)\|_\alpha \geq \|\mathbf{g}_P(x)\|_\alpha$. Based on Definition 3, we obtain the following lemma.

Lemma 1. *For a subset P of the parameter set Θ, we have $\|C_c^\alpha(\mathbf{g}_P(x))\|_\alpha \geq \|[C_c^\alpha(\mathbf{g}_\Theta(x))]_P\|_\alpha$.*

Due to lack of space, we omit the proofs of all lemmas which appear in Appendix A.

4.3 Partition-wise Gradient Magnitude

We show the relationship of the loss reduction by updating a parameter and its gradient magnitude.

Lemma 2. *Assume that we update only a single model parameter by using the gradient descent. If the learning rate of the gradient descent is small enough that the gradient does not change in every parameter value between the current and updated values of the parameter with a gradient descent step, the reduction of loss is proportional to the square of the gradient magnitude of the parameter.*

By Lemma 2, the parameters with large gradients contribute more to accuracy by reducing the loss. To reduce the amount of inserted noise into gradient magnitudes, instead of using the granularity of individual parameters, we select the parameters using the granularity of parameter partitions each of which is represented by a single weight matrix and its corresponding bias vector. Since the gradient norm of a partition tends to increase as the partition size increases, we select the partitions P_i with the largest partition-wise normalized gradient norms of the sum vector of per-example gradients, denoted by $\mu_{S,P_i}(D)$, which is defined as follows.

Definition 6. *For the norm order $\alpha \in \{1, 2\}$, a training data D and a parameter partition set $S = \{P_1, P_2, \ldots, P_{|S|}\}$ of the parameter set Θ, the α-norm of the gradient of a partition P_i is*

$\|\sum_{x\in D}\mathbf{g}_{P_i}(x)\|_\alpha = (\sum_{i=1}^{|P_i|}|\sum_{x\in D} g_{p_i}(x)|^\alpha)^{1/\alpha}$ *and* $\mu_{S,P_i}(D)$ *is defined as:*

$$\mu_{S,P_i}(D) = \frac{\|\sum_{x\in D}\mathbf{g}_{P_i}(x)\|_\alpha}{|P_i|^{1/\alpha}} = \left(\frac{\sum_{i=1}^{|P_i|}|\sum_{x\in D} g_{p_i}(x)|^\alpha}{|P_i|}\right)^{1/\alpha}.$$

4.4 Computing Differentially Private PGMs

We provide the definitions to describe PGMs.

Definition 7. *Consider a training data D, a partition set $S=\{P_1,\ldots,P_{|S|}\}$ and a gradient function $g: D\rightarrow\mathbb{R}^{|\bigcup_{P\in S}P|}$. For $\alpha=1,2$, the α-norm of the projected sum vector of clipped $g(x)$ with every $x\in D$ onto the partition P_i is denoted by $f(D,\kappa,g,P_i)$ and defined by*

$$f(D,\kappa,g,P_i) = \left\|\left[\sum_{x\in D} C_\kappa^\alpha(g(x))\right]_{P_i}\right\|_\alpha.$$

For an example $x\in D$, $\bar{\mathbf{g}}_S^\alpha(x)$ is the concatenation of the normalized gradient of each partition in S by the partition size. That is,

$$\bar{\mathbf{g}}_S^\alpha(x) = \bar{\mathbf{g}}_{P_1}^\alpha(x)\oplus\cdots\oplus\bar{\mathbf{g}}_{P_{|S|}}^\alpha(x)$$

where $\bar{\mathbf{g}}_{P_i}^\alpha(x) = \mathbf{g}_{P_i}(x)/|P_i|^{1/\alpha}$ and $\oplus$ is the concatenation operator.

For the clipping threshold c, we define $\bar{c} = c/(\frac{\sum_{P\in S}|P|}{|S|})^{1/\alpha}$. In addtion, for a vector $v=\langle v_1,\ldots,v_d\rangle$, we define $\mathrm{abs}(v)=\langle|v_1|,\ldots,|v_d|\rangle$.

By using Definition 7, we next present three different variations of PGMs.

Definition 8. *For a training data D and a partition set $S=\{P_1,P_2,\ldots,P_{|S|}\}$, we define*

- $\mu_{S,P_i}^c(D) = f(D,c,\mathbf{g}_{(\bigcup_{P\in S}P)}(x),P_i)/|P_i|^{1/\alpha}$
  $= \|[\sum_{x\in D} C_c^\alpha(\mathbf{g}_{(\bigcup_{P\in S}P)}(x))]_{P_i}\|_\alpha/|P_i|^{1/\alpha}$
- $\mu_{S,P_i}^n(D) = f(D,\bar{c},\bar{\mathbf{g}}_S^\alpha(x),P_i)$
  $= \left\|\left[\sum_{x\in D} C_{\bar{c}}^\alpha(\bar{\mathbf{g}}_S^\alpha(x))\right]_{P_i}\right\|_\alpha$
- $\mu_{S,P_i}^a(D) = f(D,\bar{c},\mathrm{abs}(\bar{\mathbf{g}}_S^\alpha(x)),P_i)$
  $= \|\sum_{x\in D}[C_{\bar{c}}^\alpha(\mathrm{abs}(\bar{\mathbf{g}}_S^\alpha(x)))]_{P_i}\|_\alpha$

To bound the variance of inserted noise into PGMs, $\mu_{S,P_i}^c(D)$ clips every per-example gradient when computing $\mu_{S,P_i}(D)$ in Definition 6. While $\mu_{S,P_i}^c(D)$ normalizes the partition-wise norm of the sum vector of clipped per-example gradients with a clipping threshold c, $\mu_{S,P_i}^n(D)$ clips the normalized per-example gradients with the clipping threshold $\bar{c}$ and computes the partition-wise norm of the sum vector of clipped per-example gradients. Moreover, $\mu_{S,P_i}^a(D)$ simply takes absolute value in every dimension of per-example gradients before clipping per-example gradients when computing $\mu_{S,P_i}^n(D)$.

By using $\mu_{S,P_i}^c(D)$, $\mu_{S,P_i}^n(D)$ and $\mu_{S,P_i}^a(D)$, we define $\mu_S^c(D)$, $\mu_S^n(D)$ and $\mu_S^a(D)$, respectively.

Definition 9. *For a training data D and a partition set $S=\{P_1,P_2,\ldots,P_{|S|}\}$, we define $\mu_S^a(D) = \langle\mu_{S,P_1}^a(D),\ldots,\mu_{S,P_{|S|}}^a(D)\rangle$. Similarly, we define $\mu_S^c(D)$ and $\mu_S^a(D)$.*

We next provide the sensitivities of μ_S^c, μ_S^n and μ_S^a (i.e., $\Delta\mu_S^c$, $\Delta\mu_S^n$ and $\Delta\mu_S^a$).

Lemma 3. *Given a partition set $S=\{P_1,P_2,\ldots,P_{|S|}\}$, for $\alpha=1,2$, we have*

- $\Delta\mu_S^c = c/(\min_{P\in S}|P|)^{1/\alpha}$
- $\Delta\mu_S^n = \Delta\mu_S^a = \bar{c} = c/(\frac{\sum_{P\in S}|P|}{|S|})^{1/\alpha}$

The standard deviation of noise is proportional to the sensitivity (Mironov, 2017). If there is a small partition in S, $\Delta\mu_S^c$ is large. To reduce the sensitivity of PGMs, $\mu_{S,P_i}^n(D)$ first normalizes the gradient of each example by partition sizes to obtain $\bar{\mathbf{g}}_S^\alpha(x)$. Thus, for each partition P_i, the PGM computed from $\bar{\mathbf{g}}_S^\alpha(x)$ (i.e., $\|[\sum_{x\in D} C_c^\alpha(\bar{\mathbf{g}}_S^\alpha(x))]_{P_i}\|_\alpha$) becomes much smaller and the noise effect can increase. However, if we set the clipping threshold for $\bar{\mathbf{g}}_S^\alpha(x)$ to $\bar{c} = c/((\sum_{P\in S}|P|)/|S|)^{1/\alpha}$, since $\min_{P\in S}|P| \leq (\sum_{P\in S}|P|)/|S|$ while the PGM can be similar to $\mu_{S,P_i}^c(D)$, the noise effect can be reduced by Lemma 3.

We found by experiments that the selected partitions based on the PGM using Definition 6 are similar whether we take an absolute value or not for each element in a per-example gradient. If we use $\mu_{S,P_i}^a(D)$ instead of $\mu_{S,P_i}^n(D)$, the PGMs become larger while the variance of inserted noise is still the same by Lemma 3. Thus, the noise effect using $\mu_{S,P_i}^a(D)$ can be reduced than using $\mu_{S,P_i}^n(D)$.

To compute noisy PGMs, we propose the DP-MG, DP-MGN and DP-MGNA methods that utilize $\mu_{S,P_i}^c(D)$, $\mu_{S,P_i}^n(D)$ and $\mu_{S,P_i}^a(D)$, respectively. To get the noisy $\mu_{S,P_i}^a(D)$, we compute $\mu_{S,P_i}^a(D)+z$ where z is noise sampled from $\mathcal{N}(0,\sigma_{\mathrm{ps}}^2(\Delta\mu_S^a)^2)$ where the noise multiplier σ_{ps} is computed by the method in (Mironov et al., 2019; Balle et al., 2020) such that the total privacy budget consumption does not exceed ϵ. For $\mu_{S,P_i}^c(D)$ and $\mu_{S,P_i}^n(D)$, noise z can be similarly sampled.

Discussion: As α increases, the α-norm of a vector is largely affected by a few large values in the vec-

tor. Thus, we expect that 1-norm is more desirable than 2-norm to select parameter partitions by using $\mu^c_{S,P_i}(D)$, $\mu^n_{S,P_i}(D)$ and $\mu^a_{S,P_i}(D)$.

4.5 Selecting Parameter Partitions

For ease of presentation, we present only the partition selection using the DP-MGNA method.

Given a partition set $S = \{P_1, P_2, \ldots, P_{|S|}\}$ and a maximum ratio γ of parameters to be unfreezed, we want to select a maximum-sized partition set $S^* = \{P_{i_1}, P_{i_2}, ..., P_{i_{|S^*|}}\}$, which consists the partitions with the largest $\mu^a_{S,P_i}(D)$s, such that $\sum_{j=1}^{|S^*|} |P_{i_j}| \leq \gamma|\Theta|$.

When computing $\mu^a_{S,P_i}(D)$, if we exclude the partitions with large noisy $\mu^a_{S,P_i}(D)$s, by Lemma 1, for each remaining partition P_i, we can reduce the effect of added noise to $\mu^a_{S,P_i}(D)$ since $\mu^a_{S,P_i}(D)$ is at least that before excluding the partitions. Moreover, we utilize a sample X of D to compute the noisy $\mu^a_{S,P_i}(X)$ quickly.

To select the parameter partitions, we repeatedly perform the following steps. At the t-th iteration, we (1) obtain a set of examples X_t by sampling each example in D with probability q_{ps}, (2) compute the noisy $\mu^a_{S,P_i}(X_t)$s of unselected partitions P_i yet by the DP-MGNA method in Section 4.4, (3) estimate the true $\mu^a_{S,P_i}(D)$s of partitions P_i from the computed noisy $\mu^a_{S,P_i}(X_t)$s so far by a maximum likelihood estimation (MLE) method, and (4) select the partitions with the largest estimated $\mu^a_{S,P_i}(D)$s among remaining partitions P_i.

Estimating the true $\mu^a_{S,P_i}(D)$s: For a disjoint partition set $S = \{P_1, P_2, \ldots, P_M\}$ of Θ, let S_t be the set of selected partitions until the t-th iteration and R_t be the set of unselected remaining partitions immediately after the t-th iteration (i.e., $S - S_t$). At the beginning of the t-th iteration, S_{t-1} and R_{t-1} are the sets of previously selected partitions and unselected remaining partitions, respectively.

Let v_i be the unknown true $\mu^a_{S,P_i}(D)$ of a partition P_i and X_t be the sampled example set at the t-th iteration. We assume $\mu^a_{R_{t-1},P_i}(X_t)/v_i = \lambda_t$ regardless of the partition P_i. Then, the unknown true $\mu^a_{R_{t-1},P_i}(X_t)$ is $\lambda_t v_i$. Note that when $t = 1$, since $R_0 = S$, we assume $\lambda_1 = q_{\text{ps}}$ where q_{ps} is the sampling rate. Moreover, let $\tilde{v}_{t,i}$ be the observed noisy $\mu^a_{R_{t-1},P_i}(X_t)$ for a partition P_i, and $\tilde{v}_t$ be $\{\tilde{v}_{t,i} \mid P_i \in R_{t-1}\}$.

We propose a maximum likelihood estimation (MLE) method to estimate $v_1, \ldots, v_M$ and $\lambda_1, \lambda_2, \ldots, \lambda_t$ by maximizing the likelihood of noisy PGMs $\tilde{v}_1, \tilde{v}_2, \ldots, \tilde{v}_t$ until the t-th iteration. Since the DP-MGNA method computes $\tilde{v}_{t,i}$ by

$$\tilde{v}_{t,i} = \mu^a_{R_{t-1},P_i}(X_t) + z = \lambda_t v_i + z \qquad (1)$$

where z is Guassian noise sampled from $\mathcal{N}(0, \sigma^2_{\text{ps}}(\Delta\mu^a_S)^2)$, $\tilde{v}_{t,i}$ has the following probability density function:

$$f(\tilde{v}_{t,i}) = \frac{1}{\sigma_{\text{ps}}\Delta_t\sqrt{2\pi}} \exp\Big(-\frac{1}{2}\Big(\frac{\tilde{v}_{t,i} - \lambda_t v_i}{\sigma_{\text{ps}}\Delta_t}\Big)^2\Big)$$

where $\Delta_t = \Delta\mu^a_{R_{t-1}}$.

Let $\ell(\tilde{v}_1, \ldots, \tilde{v}_t)$ be the log-likelihood of noisy PGMs $\tilde{v}_1, \ldots, \tilde{v}_t$. Then, we have

$$\begin{aligned}\ell(\tilde{v}_1, \ldots, \tilde{v}_t) &= \sum_{j=1}^{t} \sum_{P_i \in R_{j-1}} \log f(\tilde{v}_{j,i}) \\ &= -\sum_{j=1}^{t} \sum_{P_i \in R_{j-1}} \log(\sigma_{\text{ps}}\Delta_j\sqrt{2\pi}) \\ &\quad - \sum_{j=1}^{t} \frac{1}{2(\sigma_{\text{ps}}\Delta_j)^2} \sum_{P_i \in R_{j-1}} (\tilde{v}_{j,i} - \lambda_j v_i)^2.\end{aligned}$$

We next show λ_js to maximize $\ell(\tilde{v}_1, \ldots, \tilde{v}_t)$ of noisy PGMs when v_is are fixed.

Lemma 4. *Given $v_1, \ldots, v_M$, the log-likelihood $\ell(\tilde{v}_1, \ldots, \tilde{v}_t)$ of noisy PGMs is maximized by $\lambda_j = (\sum_{P_i \in R_{j-1}} \frac{\tilde{v}_{j,i} v_i}{\Delta_j^2})/(\sum_{P_i \in R_{j-1}} \frac{v_i^2}{\Delta_j^2})$ with $1 \leq j \leq t$.*

We next present v_is to maximize $\ell(\tilde{v}_1, \ldots, \tilde{v}_t)$ when λ_ts are fixed.

Lemma 5. *Given $\lambda_1 = q_{\text{ps}}$, $\lambda_2, \ldots, \lambda_t$, the log-likelihood $\ell(\tilde{v}_1, \ldots, \tilde{v}_t)$ of noisy PGMs is maximized by $v_i = (\sum_{j \in \mathcal{T}_{t,i}} \frac{\tilde{v}_{j,i}\lambda_j}{\Delta_j^2})/(\sum_{j \in \mathcal{T}_{t,i}} \frac{\lambda_j^2}{\Delta_j^2})$ where $\mathcal{T}_{t,i} = \{j \mid P_i \in R_{j-1} \wedge 1 \leq j \leq t\}$ with $1 \leq i \leq M$.*

We next derive v_i's variance of a partition $P_i \in R_t$.

Lemma 6. *Given $\lambda_1, \ldots, \lambda_t$, the variance of v_i, denoted by $\text{Var}(v_i)$, is $\sigma^2_{\text{ps}}/(\sum_{j=1}^{t} \frac{\lambda_j^2}{\Delta_j^2})$ for every partition $P_i \in R_t$ at the t-th iteration.*

By using Lemmas 4 and 5, we propose the *PGM-EST* (PGM ESTimation) algorithm which estimates the true $\mu^a_{S,P_i}(D)$ (i.e., v_i) of every partition P_i in the partition set $S = \{P_1, P_2, \ldots, P_M\}$ of Θ.

The *PGM-EST* algorithm: It takes $\tilde{v}_1, \ldots, \tilde{v}_t$, $\Delta_1, \ldots, \Delta_t$ as inputs and finds $v_1, \ldots, v_M$ by maximizing $\ell(\tilde{v}_1, \ldots, \tilde{v}_t)$. If $t = 1$ (i.e., the first iteration), we immediately return $\tilde{v}_{1,i}/\lambda_1$ $(= \tilde{v}_{1,i}/q_{\text{ps}})$ of all partitions. If $t \geq 2$, we first initialize

$\lambda_1, \ldots, \lambda_t$ to 1. Then, for a given number of iterations J, we repeatedly perform the following steps J times: (1) calculate $v_1, \ldots, v_M$ by maximizing $\ell(\tilde{v}_1, \ldots, \tilde{v}_t)$ with the current values of $\lambda_1, \ldots, \lambda_t$ by Lemma 5, and (2) update $\lambda_1, \ldots, \lambda_t$ by maximizing $\ell(\tilde{v}_1, \ldots, \tilde{v}_t)$ with the current values of $v_1, \ldots, v_M$ by Lemma 4. After finishing J iterations, we return $v_1, \ldots, v_M$. The details of the *PGM-EST* algorithm are in Appendix C.

Selecting partitions with estimated $\mu^a_{S,P_i}(D)$s: We present the *PSEL* (Partition SELection) algorithm that iteratively selects the partitions based on the estimated v_is by the *PGM-EST* algorithm.

We want to choose only the partitions P_is whose estimated v_is are significantly larger than the maximum of v_js of the partitions P_js that are expected to be freezed based on their v_js. We first compute the partition set S_{freeze} whose partitions P_js are expected to be freezed based on their v_js. The partition set S_{freeze} is obtained by selecting the partitions P_js with the smallest v_js in S until selecting at least $(1-\gamma)|\Theta|$ parameters. Then, we compute the maximum value, denoted by μ_{thres}, of v_js of the partitions P_js in S_{freeze}. We next choose the partitions P_is such that $v_i > \mu_{\text{thres}} + \nu\sqrt{\text{Var}(v_i)}$, where the gap coefficient ν is a user-defined value, and the total number of parameters in the selected partitions does not exceed $t \cdot \gamma \cdot |\Theta|/T$.

The *PSEL* algorithm: Let S be the disjoint partition set $\{P_1, P_2, \ldots, P_M\}$ of Θ. We first set the remaining partition set R_0 to S and set the selected partition set S_1 to $\{\}$. Then, for each t-th iteration with $t = 1, \ldots, T$, we repeatedly perform the following steps.

1. Obtain a set of examples X_t by sampling each example in D with probability q_{ps}.
2. For each $P_i \in R_{t-1}$, get $\tilde{v}_{t,i}$ by Equation (1).
3. For $i = 1, \ldots, M$, set v_i to the estimated true $\mu^a_{R_{t-1},P_i}(D)$ by the *PGM-EST* algorithm.
4. Compute $\text{Var}(v_i)$ for $P_i \in R_{t-1}$ by Lemma 6.
5. Set the PGM threshold μ_{thres} to v_j where $j = \text{argmin}_{j'} \sum_{k=1}^{j'} |P_k| \geq (1-\gamma)|\Theta|$.
6. When $t < T$, add P_is with the largest v_is satisfying $v_i > \mu_{\text{thres}} + \nu\sqrt{\text{Var}(v_i)}$ in R_{t-1} to S_t as long as $\sum_{P_i \in S_t} |P_i| \leq t \cdot \gamma \cdot |\Theta|/T$.
7. When $t = T$, we add P_is with the largest v_is to S_t as long as the total number of parameters in the selected partitions does not exceed $\gamma \cdot |\Theta|$.
8. Set the remaining partition set R_t to $R_{t-1} - S_t$.

Finally, we return the set of parameters in S_T. The details of the *PSEL* algorithm are in Appendix C.

4.6 Computing the Noise Multiplier

We present the *ComputeSigma* algorithm that computes the noise multipliers σ_{SGD} and σ_{ps} for the DP-SGD and *PSEL* algorithms, respectively, such that the total privacy budget used by the DP-SGD and *PSEL* algorithms does not exceed ϵ. It first computes the minimum noise multiplier σ_{SGD} for the DP-SGD algorithm such that the consumed privacy budget by the DP-SGD algorithm does not exceed $r_\epsilon \cdot \epsilon$ where r_ϵ is the privacy budget ratio for the DP-SGD algorithm. By using the obtained noise multiplier σ_{SGD}, we next find the minimum noise multiplier σ_{ps}, which will be used by the *PSEL* algorithm, such that the total privacy budget used by the DP-SGD algorithm and the *PSEL* algorithm does not exceed ϵ. To compute the minimum noise multipliers σ_{SGD} or σ_{ps}, we utilize the library *opacus* (Yousefpour et al., 2021), which provides the procedures to compute the privacy budget consumption ϵ by the composition of algorithms based on the RDP accountant method in (Mironov et al., 2019; Balle et al., 2020). The details of the *ComputeSigma* algorithm are presented in Appendix C.

4.7 The *DP-FROST* Algorithm

We present the *DP-FROST* algorithm which splits model parameters into partitions, computes the noise multipliers σ_{SGD} and σ_{ps} by the *ComputeSigma* algorithm, selects the partitions to be unfrozen by the *PSEL* algorithm and updates only the parameters in the selected partitions by the DP-SGD algorithm. Unless otherwise stated, the DP-FROST algorithm uses the DP-MGNA method in Section 4.4 with $\alpha = 1$ since it is more desirable than $\alpha = 2$ to select parameter partitions as discussed in Section 4.4.

Its pseudocode is provided in Algorithm 1. We first split the model parameters into a set of partitions $S = \{P_1, P_2, \ldots, P_M\}$ (line 1). We next compute the noise multipliers σ_{SGD} and σ_{ps} for the DP-SGD algorithm (Abadi et al., 2016) and the *PSEL* algorithm in Algorithm 4, respectively, by calling the *ComputeSigma* algorithm (line 2). Then, the parameter set P^* to be fine-tuned is obtained by invoking the *PSEL* algorithm with the noise multiplier σ_{ps} (line 3). Furthermore, we set T_{SGD} and q_{SGD} to the number of parameter updates and the probability to sample each example, respectively (lines 4-5). We update only the parameters in P^* by repeating the following steps T_{SGD} times (line 7-10): (1) draw a mini-batch X

Algorithm 1 DP-FROST

Input: A dataset D, a privacy budget ϵ, a failure constant δ, a batch size B, the number of epochs E, a budget ratio r_ϵ, a clipping threshold c, a norm order α, a budget tolerance τ, the number of iterations during parameter selection T, the unfreezing ratio γ, a gap coefficient ν, the sampling rate of parameter selection q_{ps}

1: Create the set of disjoint parameter partitions $S = \{P_1, P_2, \ldots, P_M\}$
2: $\sigma_{\text{SGD}}, \sigma_{\text{ps}} \leftarrow$ *ComputeSigma*$(D, \epsilon, \delta, B, E, r_\epsilon, \tau, T, q_{\text{ps}})$
3: $P^* \leftarrow$ *PSEL*$(D, S, \sigma_{\text{ps}}, \alpha, T, \gamma, \nu, q_{\text{ps}})$
4: $T_{\text{SGD}} \leftarrow \lfloor E \cdot |D|/B \rfloor$
5: $q_{\text{SGD}} \leftarrow B/|D|$
6: **for** $t \leftarrow 1$ **to** T_{SGD} **do**
7: Draw a mini-batch X by sampling each example with probability q_{SGD}
8: Sample noise z from $\mathcal{N}(0, \sigma^2_{\text{SGD}} c^2 \mathbf{I})$
9: Compute $\tilde{\mathcal{G}}^c_{P^*}(X) = \frac{1}{B}(\sum_{x \in X} C^2_c(\mathbf{g}_{P^*}(x)) + z)$
10: *Update*$(P^*, \tilde{\mathcal{G}}^c_{P^*}(X))$
11: **return** Θ

(line 7), (2) compute the noisy gradient $\tilde{\mathcal{G}}^c_{P^*}(X)$ of the mini-batch X by the DP-SGD algorithm with the noise multiplier σ_{SGD} (lines 8-9), and (3) update the parameters in P^* by the *Update* procedure which utilizes an optimizer, such as Adam (Kingma and Ba, 2015) or AdamW (Loshchilov and Hutter, 2019) (line 10).

To compute the noisy gradient of a mini-batch X by the DP-SGD algorithm (lines 8-9), we clip the per-example gradient $\mathbf{g}_{P^*}(x)$ of each example x, insert noise z sampled from $\mathcal{N}(0, \sigma^2_{\text{SGD}} c^2 \mathbf{I})$ to the sum of clipped gradients (i.e., $\sum_{x \in X} C^2_c(\mathbf{g}_{P^*}(x))$) and divide the noisy gradient by the batch size B. As a result, for a mini-batch X, the DP-SGD algorithm computes the noisy gradient $\tilde{\mathcal{G}}^c_{P^*}(X)$ below.

$$\tilde{\mathcal{G}}^c_{P^*}(X) = \frac{1}{B}\left(\sum_{x \in X} C^2_c(\mathbf{g}_{P^*}(x)) + z\right).$$

The details of the *DP-FROST* algorithm are in Appendix C. We next show that the *DP-FROST* algorithm satisfies (ϵ, δ)-differential privacy.

Lemma 7. *The* DP-FROST *algorithm satisfies* (ϵ, δ)*-differential privacy (DP).*

Proof. Since DP-FROST is a composition of the sampled Gaussian mechanisms, for a failure constant δ, their sensitivities proven by Lemma 3 and fixed standard deviations of noise by the mechanisms, we can compute the privacy budget consumption ϵ' satisfying (ϵ', δ)-DP based on the RDP accountant method (Mironov et al., 2019; Balle et al., 2020). In line 2 of *DP-FROST* in Algorithm 1, we find the minimum standard deviations of noise such that the consumed privacy budget ϵ' with a failure constant δ does not exceed the given privacy budget ϵ. Since we have $\epsilon' \leq \epsilon$, *DP-FROST* satisfies (ϵ, δ)-DP by Definition 1. □

5 Experiments

We run experiments 3 times in NVIDIA RTX 3090 GPU, and report the average results. Additional details and results of experiments are in Appendix D.

Compared methods: We implement the following DP methods for fine-tuning a pre-trained model.

- **Full**: It fine-tunes all model parameters.
- **Rand**: It updates randomly selected parameters.
- **PVAL**: It is a SOTA (Luo et al., 2021) for CNN models described in Section 2.
- **LoRA**: It is a SOTA method (Li et al., 2022) for language models described in Section 2.
- **Adapter**: It is a SOTA method (Yu et al., 2022) for language models described in Section 2.
- **DP-FROST**: It is our method in Section 4.

Experimental settings: We consider privacy parameters $\epsilon = 0.5, 2, 8$ and $\delta = 1/(2|D|)$ where $|D|$ is the size of the training dataset. We compared our methods with the other methods under the same (ϵ, δ)-differential privacy which provides the same level of privacy guarantee. We consider the rank parameter values of 1, 2, 4 and 8 for LoRA by following (Li et al., 2022) and 16, 48 and 96 for Adapter by following (Yu et al., 2022). For DP-FROST and PVAL, we consider parameter ratios γ of 0.25, 0.25^2, 0.25^3 and 0.25^4. The privacy budget ratio r_ϵ of the *DP-FROST* algorithm is set to 0.9. For *PSEL* algorithm, we set the number of maximum iterations T to 5, the sampling probability q_{ps} to 0.02 and the gap coefficient ν to 5. Moreover, we set the number of maximum iterations J in the *PGM-EST* algorithm to 1,000.

In training deep learning model with differential privacy, a large batch sizes that can overflow GPU memory are typically used (Li et al., 2022). Thus, we use the gradient accumulation (Gao et al., 2021), which splits a large batch into chunks and accumulate gradients computed from the chunks,

Method	$\epsilon = 0.5$				$\epsilon = 2.0$				$\epsilon = 8.0$			
	SST2	QNLI	QQP	MNLI	SST2	QNLI	QQP	MNLI	SST2	QNLI	QQP	MNLI
Non-private	94.92	92.74	91.81	87.23	94.92	92.74	91.81	87.23	94.92	92.74	91.81	87.23
Full	88.42	82.74	82.78	78.79	87.77	84.71	85.25	81.92	89.49	86.19	86.61	83.35
Rand	88.07	82.55	80.49	74.49	88.53	84.96	83.97	78.96	89.91	86.15	85.68	81.18
PVAL	88.46	83.34	83.40	78.96	89.91	85.42	85.57	82.27	90.83	86.90	86.96	83.84
LoRA	89.76	82.40	82.96	79.09	91.21	84.29	85.21	82.13	91.28	86.63	86.41	83.61
Adapter	89.30	82.04	82.27	78.72	90.56	85.45	84.75	82.15	91.17	87.00	86.28	83.85
DP-FROST	**90.71**	**84.14**	**83.76**	**79.86**	**91.32**	**86.34**	**85.79**	**83.09**	**91.44**	**87.75**	**87.02**	**84.36**

Table 1: The accuracy of text classification

whose size is called the physical batch size (Bu et al., 2023). The physical batch size only affects the speed or memory consumption but not the accuracy of training process. We use the maximum physical batch size such that the memory consumption does not exceed the memory capacity.

Natural language processing (NLP) tasks: Following the settings in (Li et al., 2022; Yu et al., 2022), we compare with the SOTAs for differentially private fine-tuning using the pre-trained RoBERTa-Base (Liu et al., 2019) on 4 real datasets from GLUE benchmark (Wang et al., 2018).

- **SST-2**: It has 70,042 movie review sentences. The task predicts the sentiment of a sentence.
- **QNLI**: It has 115,669 question-sentence pairs. The task is to determine whether the sentence contains the answer to the question.
- **QQP**: It is a collection of 795,241 question pairs. The task is to determine whether a pair of questions are semantically equivalent.
- **MNLI**: It has 431,992 sentence pairs. The task predicts the textual entailment of a sentence pair.

We conduct experiments similarly with (Li et al., 2022). We use the AdamW optimizer (Loshchilov and Hutter, 2019) with varying its learning rate of 0.003, 0.001, 0.0003 and 0.0001 and set the batch size in SST-2 to 1024. For each of QNLI, QQP and MNLI, we set the batch size such that the ratio of the batch size to the data size is the same as that of the batch size in SST-2 to the data size. Furthermore, for each dataset, we use the default number of epochs provided in (Li et al., 2022). The numbers of epochs for SST-2, QNLI, QQP and MNLI are 3, 6, 18 and 18, respectively.

Computer vision (CV) task: Following the settings in (Bu et al., 2022), we fine-tune the pre-trained ViT-Base (Dosovitskiy et al., 2021) for image classification on the CIFAR100 dataset (Krizhevsky et al., 2009) consisting of 100 classes with 600 images per class. We set the batch size to 1,000 and the number of epochs to 3 following the setting in (Bu et al., 2022). We use the Adam optimizer (Kingma and Ba, 2015) with varying its learning rate of 0.0003, 0.001 and 0.003.

Table 2: The accuracy of image classification

Method	$\epsilon = 0.5$	$\epsilon = 2.0$	$\epsilon = 8.0$
Full	75.15	84.36	86.57
Rand	78.22	84.02	86.42
PVAL	82.52	87.19	88.65
LoRA	80.61	87.02	88.55
Adapter	75.17	84.40	86.67
DP-FROST	**83.51**	**87.31**	**88.72**

5.1 Main Results

We experiment with both NLP and CV tasks to validate the effectiveness of our proposed method.

NLP tasks: We present the text classification accuracy of the fine-tuned RoBERTa-Base in Table 1. DP-FROST outperforms all other methods in every dataset. Moreover, with decreasing privacy budget ϵ, the performance gap between DP-FROST and the other methods tends to increase. Thus, DP-FROST is effective to improve the accuracy especially for a small ϵ which is required for strong privacy protection by inserting large noise.

CV task: We provide the accuracy of the fine-tuned ViT-Base on CIFAR100 in Table 2. The ViT-Base trained by DP-FROST performs the best for $\epsilon = 0.5, 2, 8$. The results show that the proposed method also effective in the CV task.

5.2 Ablation Study

We compare the accuracy of fine-tuned models with different parameter selection methods with $\epsilon = 0.5$ and show the results in Table 3. PARAM selects the model parameters individually based on the noisy gradient magnitudes. DP-MG, DP-MGN and DP-MGNA denote the DP-FROST methods with the corresponding private PGM computation meth-

Table 3: Comparison of parameter selection methods

Method	SST-2	QNLI	QQP	MNLI
PARAM	84.40	81.44	78.03	70.59
DP-MG (SINGLE)	87.96	81.84	79.81	74.27
DP-MG	88.11	82.18	80.85	74.49
DP-MGN (SINGLE)	87.54	82.52	82.44	75.26
DP-MGN	88.65	82.36	**83.83**	79.77
DP-MGNA (SINGLE)	89.56	83.64	**83.83**	**80.35**
DP-MGNA	**90.71**	**84.14**	83.76	79.86

Table 4: Varying the number of partition selections K

K	SST-2	QNLI	QQP	MNLI
1	**90.71**	**84.14**	**83.76**	**79.86**
2	**90.71**	83.73	82.24	75.70
4	89.79	83.85	80.89	72.50

ods, respectively. Moreover, SINGLE indicates the variant of DP-FROST that selects partitions at once without using the *PSEL* algorithm. As we expected, the partition-wise selection is more effective than individual selection methods. DP-MGN significantly outperforms DP-MG on QQP due to the effectiveness of the proposed normalization. In addition, we observed additional performance improvement when we use the absolute value of each element in the gradient (DP-MGNA). Since the *PSEL* algorithm estimates the true PGMs of partitions more accurately by reducing the effect of inserted noise in the noisy PGMs of partitions, using it is better or similar to the methods without the *PSEL* algorithm (SINGLE).

5.3 Hyperparameter Analysis

Partition selections during training: While DP-FROST selects the unfreezed partitions only once before training, we study how the accuracy changes if we also change the unfreezed partitions multiple times during training. Let K and U be the numbers of partition selections and batch updates while we fine-tune the pre-trained model, respectively. At the beginning of every batch update $(k-1)\cdot\lfloor U/K \rfloor$ with $1 \leq k \leq K$, we select the unfreezed partitions by calling the *PSEL* algorithm in Section 4.5. Table 4 shows the accuracy of the fined-tuned RoBERTa-Base for $\epsilon = 0.5$. The fine-tuned model with $K=1$ (i.e., selecting once before training) outperforms those with $K=2,4$. This is because the inserted noise becomes large when the privacy budget is divided for multiple partition selections.

Training times: Table 5 shows the training times using RoBERTa-Base for $\epsilon = 0.5$. Since DP-FROST, LoRA and Adapter compute gradients of only some weight matrices, they are faster than Full and PVAL that compute gradients of all weight matrices. Although DP-FROST computes gradients over the entire weight matrices during parameter selection, since it computes the gradients of sample examples only, its parameter selection of takes at most 11 % of its total training time. DP-FROST takes up to 38 % more time than the fastest Adapter, but DP-FROST is the most accurate as in Table 1.

Table 5: The training time (secs) of text classification

Method	SST-2	QNLI	QQP	MNLI
Full	957	6798	39024	52425
PVAL	1215	7692	46818	57970
LoRA	354	3672	18180	26113
Adapter	284	2154	14850	25657
DP-FROST	367	2580	20463	27774

Table 6: Varying the sampling rate q_{ps} of *PSEL*

q_{ps}	1.0	0.1	0.05	0.02	0.01
Accuracy	91.13	91.09	90.67	90.71	90.10
Running time	2138	514	435	384	367

Varying the sampling rate q_{ps}: To see the effect of using a sample of training data to perform the parameter selection by the *PSEL* algorithm, for SST-2 with $\epsilon = 0.5$, we present the accuracy and running time (in seconds) of DP-FROST with varying the sampling probability q_{ps} in Table 6. When $q_{\text{ps}} \geq 0.02$, there is almost no difference in accuracy. However, when $q_{\text{ps}} < 0.02$, the accuracy decreases due to a small sample size.

6 Conclusion

We have studied the differentially private fine-tuning of pre-trained models with freezing model parameters. We proposed the DP-FROST method which selects important model parameters based on the gradient magnitudes of parameter partitions. The experimental results showed the effectiveness of our proposed method.

Limitations: We focus on improving the accuracies of fine-tuned models and the proposed method is not the the fastest among the compared methods and takes up to a little more time than the fastest compared method. Moreover, extending our research to the local differential privacy and federated learning is a future work.

Acknowledgments

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (RS-2023-00247438), and was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grants funded by the Korea government(MSIT) (No. 2022-0-00516, Derivation of a Differential Privacy Concept Applicable to National Statistics Data While Guaranteeing the Utility of Statistical Analysis) and (No.RS-2023-00261068, Development of Lightweight Multimodal Anti-Phishing Models and Split-Learning Techniques for Privacy-Preserving Anti Phishing). This research was also supported by the MSIT(Ministry of Science and ICT), Korea, under the Convergence security core talent training business support program(IITP2024-RS-2024-00423071) supervised by the IITP(Institute of Information & Communications Technology Planning & Evaluation).

References

Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 308–318, New York, NY, USA. Association for Computing Machinery.

Tom M Apostol. 1967. One-variable calculus, with an introduction to linear algebra.

Borja Balle, Gilles Barthe, Marco Gaboardi, Justin Hsu, and Tetsuya Sato. 2020. Hypothesis testing interpretations and renyi differential privacy. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 2496–2506. PMLR.

Zhiqi Bu, Jialin Mao, and Shiyun Xu. 2022. Scalable and efficient training of large convolutional neural networks with differential privacy. In *Advances in Neural Information Processing Systems*, volume 35, pages 38305–38318. Curran Associates, Inc.

Zhiqi Bu, Yu-Xiang Wang, Sheng Zha, and George Karypis. 2023. Differentially Private Optimization on Large Model at Small Cost. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 3192–3218. PMLR.

Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. 2019. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *Proceedings of the 28th USENIX Conference on Security Symposium*, SEC'19, page 267–284, USA. USENIX Association.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom B Brown, Dawn Song, Ulfar Erlingsson, et al. 2021. Extracting training data from large language models. In *USENIX Security Symposium*, volume 6.

Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. 2022. *Introduction to algorithms*. MIT press.

Daniel Daners. 2008. Introduction to functional analysis. *The University of Sydney*.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *Theory of Cryptography*, pages 265–284, Berlin, Heidelberg. Springer Berlin Heidelberg.

Cynthia Dwork and Aaron Roth. 2013. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–487.

Luyu Gao, Yunyi Zhang, Jiawei Han, and Jamie Callan. 2021. Scaling deep contrastive learning batch size under memory limited setup. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, pages 316–321, Online. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images.

Xuechen Li, Florian Tramer, Percy Liang, and Tatsunori Hashimoto. 2022. Large Language Models Can Be Strong Differentially Private Learners. In *International Conference on Learning Representations*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Zelun Luo, Daniel J Wu, Ehsan Adeli, and Li Fei-Fei. 2021. Scalable Differential Privacy with Sparse Network Finetuning. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5057–5066.

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

Ilya Mironov, Kunal Talwar, and Li Zhang. 2019. Rényi Differential Privacy of the Sampled Gaussian Mechanism. *CoRR*, abs/1908.1.

Wahbeh Qardaji, Weining Yang, and Ninghui Li. 2013. Differentially private grids for geospatial data. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pages 757–768.

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

Reza Shokri and Vitaly Shmatikov. 2015. Privacy-Preserving Deep Learning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 1310–1321, New York, NY, USA. Association for Computing Machinery.

Zongkun Sun, Yinglong Wang, Minglei Shu, Ruixia Liu, and Huiqi Zhao. 2019. Differential privacy for data and model publishing of medical data. *Ieee Access*, 7:152103–152114.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Ashkan Yousefpour, Igor Shilov, Alexandre Sablayrolles, Davide Testuggine, Karthik Prasad, Mani Malek, John Nguyen, Sayan Ghosh, Akash Bharadwaj, Jessica Zhao, Graham Cormode, and Ilya Mironov. 2021. Opacus: User-friendly differential privacy library in PyTorch. *arXiv preprint arXiv:2109.12298*.

Da Yu, Saurabh Naik, Arturs Backurs, Sivakanth Gopi, Huseyin A Inan, Gautam Kamath, Janardhan Kulkarni, Yin Tat Lee, Andre Manoel, Lukas Wutschitz, Sergey Yekhanin, and Huishuai Zhang. 2022. Differentially Private Fine-tuning of Language Models. In *International Conference on Learning Representations*.

Qiuchen Zhang, Jing Ma, Yonghui Xiao, Jian Lou, and Li Xiong. 2020. Broadening differential privacy for deep learning against model inversion attacks. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 1061–1070.

A The Proofs of Lemmas

Lemma 1. *For a subset P of the parameter set Θ, we have $\|C_c^\alpha(\mathbf{g}_P(x))\|_\alpha \geq \|[C_c^\alpha(\mathbf{g}_\Theta(x))]_P\|_\alpha$.*

Proof. For a subset P of the parameter set Θ, we have

$$\sum_{p\in P}\left|\frac{\partial\mathcal{L}(x)}{\partial p}\right|^\alpha \leq \sum_{p\in\Theta}\left|\frac{\partial\mathcal{L}(x)}{\partial p}\right|^\alpha. \tag{2}$$

Since the α-norm of a vector $v=\langle v_1, v_2, \ldots, v_d\rangle$, denoted by $\|v\|_\alpha$, is $\|v\|_\alpha = (\sum_{i=1}^d |x_i|^\alpha)^{1/\alpha}$ and we have $\|\mathbf{g}_P(x)\|_\alpha \leq \|\mathbf{g}_\Theta(x)\|_\alpha$ by Equation (2), we can derive

$$\begin{aligned}
&\|C_c^\alpha(\mathbf{g}_P(x))\|_\alpha \\
&= \|\min(1, c/\|\mathbf{g}_P(x)\|_\alpha)\cdot\mathbf{g}_P(x)\|_\alpha \\
&\qquad\qquad \text{(by Definition 3)} \\
&\geq \|\min(1, c/\|\mathbf{g}_\Theta(x)\|_\alpha)\cdot\mathbf{g}_P(x)\|_\alpha \\
&= \|\min(1, c/\|\mathbf{g}_\Theta(x)\|_\alpha)\cdot[\mathbf{g}_\Theta(x)]_P\|_\alpha \\
&= \|[\min(1, c/\|\mathbf{g}_\Theta(x)\|_\alpha)\cdot\mathbf{g}_\Theta(x)]_P\|_\alpha \\
&= \|[C_c^\alpha(\mathbf{g}_\Theta(x))]_P\|_\alpha. \qquad \text{(by Definition 3)}
\end{aligned}$$

Thus, $\|C_c^\alpha(\mathbf{g}_P(x))\|_\alpha \geq \|[C_c^\alpha(\mathbf{g}_\Theta(x))]_P\|_\alpha$ holds. □

Lemma 2. *Assume that we update only a single model parameter by using the gradient descent. If the learning rate of the gradient descent is small enough that the gradient does not change in every parameter value between the current and updated values of the parameter with a gradient descent step, the reduction of loss is proportional to the square of the gradient magnitude of the parameter.*

Proof. Let p' be the current value of parameter p and p'' be the updated value of the parameter p after a gradient descent step with a learning rate η (i.e., $p''=p'-\eta\cdot\frac{\partial\mathcal{L}(X)}{\partial p}\Big|_{p=p'}$). Since the gradient does not change when the value of the parameter p is between p' and p'' in the gradient descent step, $\frac{\partial\mathcal{L}(X)}{\partial p}$ is constant. By the fundamental theorem of calculus (Apostol, 1967), we obtain

$$\begin{aligned}
&\mathcal{L}(X)|_{p=p'} - \mathcal{L}(X)|_{p=p''} \\
&= \int_{p''}^{p'}\frac{\partial\mathcal{L}(X)}{\partial p}dp = (p'-p'')\cdot\frac{\partial\mathcal{L}(X)}{\partial p}\Big|_{p=p'} \\
&= \eta\cdot\left(\frac{\partial\mathcal{L}(X)}{\partial p}\Big|_{p=p'}\right)^2. \\
&\qquad\qquad \text{(since } p''=p'-\eta\cdot\tfrac{\partial\mathcal{L}(X)}{\partial p}\Big|_{p=p'}\text{)}
\end{aligned}$$

□

Lemma 3. *Given a partition set $S = \{P_1, P_2, \ldots, P_{|S|}\}$, for $\alpha = 1, 2$, we have*

- $\Delta\mu_S^c = c/(\min_{P\in S}|P|)^{1/\alpha}$
- $\Delta\mu_S^n = \Delta\mu_S^a = \bar{c} = c/(\frac{\sum_{P\in S}|P|}{|S|})^{1/\alpha}$

Proof. We prove (1) $\Delta\mu_S^c = c/(\min_{P\in S}|P|)^{1/\alpha}$, (2) $\Delta\mu_S^n = \bar{c} = c/(\frac{\sum_{P\in S}|P|}{|S|})^{1/\alpha}$ and (3) $\Delta\mu_S^a = \bar{c} = c/(\frac{\sum_{P\in S}|P|}{|S|})^{1/\alpha}$.

(1) The proof of $\Delta\mu_S^c = c/(\min_{P\in S}|P|)^{1/\alpha}$: Consider a pair of neighboring datasets D and D' which are identical except that D' has an additional example x' (i.e., $D' - D = \{x'\}$). By Definition 2, the sensitivity $\Delta\mu_S^c$ is $\max_{D,D'}\|\mu_S^c(D) - \mu_S^c(D')\|_2$. We next show that $\Delta\mu_S^c = c/(\min_{P\in S}|P|)^{1/\alpha}$ by proving (a) $c/(\min_{P\in S}|P|)^{1/\alpha} \leq \Delta\mu_S^c$ and (b) $\Delta\mu_S^c \leq c/(\min_{P\in S}|P|)^{1/\alpha}$.

(1-a) The proof of $c/(\min_{P\in S}|P|)^{1/\alpha} \leq \Delta\mu_S^c$: Since $\Delta\mu_S^c$ is the maximum value of $\|\mu_S^c(D) - \mu_S^c(D')\|_2$ for every pair of neighboring datasets D and D', we have $\Delta\mu_S^c = \max_{D,D'}\| \mu_S^c(D) - \mu_S^c(D')\|_2 \geq \|\mu_S^c(D'') - \mu_S^c(D''')\|_2$ for any pair of neighboring datasets D'' and D'''. Thus, let $D'' = \{\}$ and $D''' = \{x'\}$ where x' is an example.

$$
\begin{aligned}
\Delta\mu_S^c &= \max_{D,D'}\|\mu_S^c(D) - \mu_S^c(D')\|_2 \\
&\geq \|\mu_S^c(D'') - \mu_S^c(D''')\|_2 \\
&= \|\mu_S^c(\{\}) - \mu_S^c(\{x'\})\|_2 \\
&= \|\mu_S^c(\{x'\})\|_2 \qquad \text{(since } \mu_S^c(\{\}) = 0\text{)} \\
&= \Big(\sum_{i=1}^{|S|}(\mu_{S,P_i}^c(\{x'\}))^2\Big)^{1/2} \qquad \text{(by Definition 9)}
\end{aligned}
$$

Since for any $P_i \in S$, we have

$$
\begin{aligned}
&\Big(\sum_{i=1}^{|S|}(\mu_{S,P_i}^c(\{x'\}))^2\Big)^{1/2} \\
&\geq ((\mu_{S,P_i}^c(\{x'\}))^2)^{1/2} = \mu_{S,P_i}^c(\{x'\})
\end{aligned}
$$

and $\mu_{S,P_i}^c(D) = \|[\sum_{x\in D} C_c^\alpha(\mathbf{g}_{(\bigcup_{P\in S}P)}(x))]_{P_i}\|_\alpha/|P_i|^{1/\alpha}$ by Definition 8, we obtain the following

$$
\begin{aligned}
&\Delta\mu_S^c \geq \mu_{S,P_{\min}}^c(\{x'\}) \\
&= \|[C_c^\alpha(\mathbf{g}_{(\bigcup_{P\in S}P)}(x))]_{P_{\min}}\|_\alpha/|P_{\min}|^{\frac{1}{\alpha}}
\end{aligned}
$$

where $P_{\min}$ is a partition $P \in S$ such that $|P_{\min}| = \min_{P\in S}|P|$. Since the above inequality holds for any possible example x' and $0 \leq \|[C_c^\alpha(\mathbf{g}_{(\bigcup_{P\in S}P)}(x))]_{P_{\min}}\|_\alpha \leq c$, we have $c/(\min_{P\in S}|P|)^{1/\alpha} \leq \Delta\mu_S^n$.

(1-b) The proof of $\Delta\mu_S^c \leq c/(\min_{P\in S}|P|)^{1/\alpha}$: For vectors a and b, the triangular inequality is

$$\|a + b\|_\alpha \leq \|a\|_\alpha + \|b\|_\alpha. \tag{3}$$

Furthermore, for vectors v and w, by letting $a = v - w$ and $b = w$ in Equation (3), we obtain $\|(v - w) + w\|_\alpha = \|v\|_\alpha \leq \|v - w\|_\alpha + \|w\|_\alpha$, which is equivalent to

$$\|v\|_\alpha - \|w\|_\alpha \leq \|v - w\|_\alpha. \tag{4}$$

By combining Definitions 2 and 9, we get

$$
\begin{aligned}
&\Delta\mu_S^c \\
&= \max_{D,D'}\Big(\sum_{i=1}^{|S|}\Big(\frac{\|[\sum_{x\in D} C_c^\alpha(\mathbf{g}_{(\bigcup_{P\in S}P)}(x))]_{P_i}\|_\alpha}{|P_i|^{\frac{1}{\alpha}}} \\
&\quad - \frac{\|[\sum_{x\in D'} C_c^\alpha(\mathbf{g}_{(\bigcup_{P\in S}P)}(x))]_{P_i}\|_\alpha}{|P_i|^{\frac{1}{\alpha}}}\Big)^2\Big)^{\frac{1}{2}} \\
&\leq \max_{D,D'}\Big(\sum_{i=1}^{|S|}\Big(\frac{1}{|P_i|^{\frac{1}{\alpha}}}\Big\|\Big[\sum_{x\in D} C_c^\alpha(\mathbf{g}_{(\bigcup_{P\in S}P)}(x))\Big]_{P_i} \\
&\quad - \Big[\sum_{x\in D'} C_c^\alpha(\mathbf{g}_{(\bigcup_{P\in S}P)}(x))\Big]_{P_i}\Big\|_\alpha\Big)^2\Big)^{\frac{1}{2}}. \\
&\qquad\qquad \text{(by Equation (4))}
\end{aligned}
$$

Since $D' - D = \{x'\}$, the above inequality becomes

$$
\begin{aligned}
&\Delta\mu_S^c \\
&\leq \max_{x'}\Big(\sum_{i=1}^{|S|}\Big(\frac{\|[C_c^\alpha(\mathbf{g}_{(\bigcup_{P\in S}P)}(x'))]_{P_i}\|_\alpha}{|P_i|^{\frac{1}{\alpha}}}\Big)^2\Big)^{\frac{1}{2}} \\
&\leq \frac{\max_{x'}(\sum_{i=1}^{|S|}(\|[C_c^\alpha(\mathbf{g}_{(\bigcup_{P\in S}P)}(x'))]_{P_i}\|_\alpha)^2)^{\frac{1}{2}}}{\min_{P\in S}|P|^{\frac{1}{\alpha}}}.
\end{aligned} \tag{5}
$$

For a vector $v = \langle v_1, v_2, ..., v_d\rangle$, we have the following inequality (Daners, 2008).

$$\Big(\sum_{i=1}^{d} v_i^2\Big)^{\frac{1}{2}} = \|v\|_2 \leq \|v\|_1 = \sum_{i=1}^{d}|v_i|. \tag{6}$$

By combining Equations (5) and (6) as well as

letting $\alpha = 1$, we can derive

$$\begin{aligned}
&\Delta\mu_S^c \\
&\leq \frac{\max_{x'}(\sum_{i=1}^{|S|}(\|[C_c^1(\mathbf{g}_{(\bigcup_{P\in S}P)}(x'))]_{P_i}\|_1)^2)^{\frac{1}{2}}}{\min_{P\in S}|P|} \\
&\leq \frac{\max_{x'}(\sum_{i=1}^{|S|}\|\|[C_c^1(\mathbf{g}_{(\bigcup_{P\in S}P)}(x'))]_{P_i}\|_1|)}{\min_{P\in S}|P|} \\
&\qquad\qquad \text{(by Equation (6))} \\
&\leq \frac{\max_{x'}(\sum_{i=1}^{|S|}\|[C_c^1(\mathbf{g}_{(\bigcup_{P\in S}P)}(x'))]_{P_i}\|_1)}{\min_{P\in S}|P|} \\
&\qquad \text{(since } \|[C_c^1(\mathbf{g}_{(\bigcup_{P\in S}P)}(x'))]_{P_i}\|_1 \geq 0\text{)} \\
&= \frac{\max_{x'}\|C_c^1(\mathbf{g}_{(\bigcup_{P\in S}P)}(x'))\|_1}{\min_{P\in S}|P|} \\
&\qquad \text{(by the definition of 1-norm)} \\
&= c/(\min_{P\in S}|P|) \\
&\qquad \text{(since } \|C_c^1(\mathbf{g}_{(\bigcup_{P\in S}P)}(x'))\|_1 \leq c\text{)}
\end{aligned}$$

On the other hand, letting $\alpha = 2$ in Equation (5) derives

$$\begin{aligned}
&\Delta\mu_S^c \\
&\leq \frac{\max_{x'}(\sum_{i=1}^{|S|}(\|[C_c^1(\mathbf{g}_{(\bigcup_{P\in S}P)}(x'))]_{P_i}\|_2)^2)^{\frac{1}{2}}}{\min_{P\in S}|P|^{\frac{1}{2}}} \\
&= \max_{x'}\frac{\|C_c^2(\mathbf{g}_{(\bigcup_{P\in S}P)}(x'))\|_2}{\min_{P\in S}|P|^{\frac{1}{2}}} \\
&\qquad \text{(by the definition of 2-norm)} \\
&= c/(\min_{P\in S}|P|)^{1/2}. \\
&\qquad \text{(since } \|C_c^2(\mathbf{g}_{(\bigcup_{P\in S}P)}(x'))\|_2 \leq c\text{)}
\end{aligned}$$

Since $c/(\min_{P\in S}|P|)^{1/\alpha} \leq \Delta\mu_S^c$ and $\Delta\mu_S^c \leq c/(\min_{P\in S}|P|\,)^{1/\alpha}$, we have

$$\Delta\mu_S^c = c/(\min_{P\in S}|P|)^{1/\alpha}.$$

(2) The proof of $\Delta\mu_S^n = \bar{c} = c/(\frac{\sum_{P\in S}|P|}{|S|})^{1/\alpha}$: Consider a pair of neighboring datasets D and D' which are identical except that D' has an additional example x' (i.e., $D' - D = \{x'\}$). By Definition 2, the sensitivity $\Delta\mu_S^n$ is $\max_{D,D'}\|\mu_S^n(D) - \mu_S^n(D')\|_2$. We next show that $\Delta\mu_S^n = \bar{c}$ by proving (a) $\Delta\mu_S^n \leq \bar{c}$ and (b) $\bar{c} \leq \Delta\mu_S^n$.

(2-a) The proof of $\Delta\mu_S^n \leq \bar{c}$: By Definitions 2 and 9, we get

$$\begin{aligned}
\Delta\mu_S^n &= \max_{D,D'}\Big(\sum_{i=1}^{|S|}\Big(\Big\|\Big[\sum_{x\in D}C_{\bar{c}}^{\alpha}(\bar{\mathbf{g}}_S^{\alpha}(x))\Big]_{P_i}\Big\|_{\alpha} \\
&\qquad - \Big\|\Big[\sum_{x\in D'}C_{\bar{c}}^{\alpha}(\bar{\mathbf{g}}_S^{\alpha}(x))\Big]_{P_i}\Big\|_{\alpha}\Big)^2\Big)^{\frac{1}{2}} \\
&\leq \max_{D,D'}\Big(\sum_{i=1}^{|S|}\Big\|\Big[\sum_{x\in D}C_{\bar{c}}^{\alpha}(\bar{\mathbf{g}}_S^{\alpha}(x))\Big]_{P_i} \\
&\qquad - \Big[\sum_{x\in D'}C_{\bar{c}}^{\alpha}(\bar{\mathbf{g}}_S^{\alpha}(x))\Big]_{P_i}\Big\|_{\alpha}^2\Big)^{\frac{1}{2}} \\
&\qquad\qquad \text{(by Equation (4))}
\end{aligned}$$

Since $D' - D = \{x'\}$, the above inequality becomes

$$\Delta\mu_S^n \leq \max_{D,D'}\Big(\sum_{i=1}^{|S|}\|[C_{\bar{c}}^{\alpha}(\bar{\mathbf{g}}_S^{\alpha}(x'))]_{P_i}\|_{\alpha}^2\Big)^{\frac{1}{2}}. \quad (7)$$

By letting $\alpha = 1$ in Equation (7), we can derive

$$\begin{aligned}
&\Delta\mu_S^n \leq \max_{D,D'}\Big(\sum_{i=1}^{|S|}\|[C_{\bar{c}}^1(\bar{\nabla}_S^1\mathcal{L}(x'))]_{P_i}\|_1^2\Big)^{\frac{1}{2}} \\
&\leq \max_{D,D'}\Big(\sum_{i=1}^{|S|}\|\|[C_{\bar{c}}^1(\bar{\nabla}_S^1\mathcal{L}(x'))]_{P_i}\|_1|\Big) \\
&\qquad\qquad \text{(by Equation (6))} \\
&\leq \max_{D,D'}\Big(\sum_{i=1}^{|S|}\|[C_{\bar{c}}^1(\bar{\nabla}_S^1\mathcal{L}(x'))]_{P_i}\|_1\Big) \\
&\qquad \text{(since } \|[C_{\bar{c}}^1(\bar{\nabla}_S^1\mathcal{L}(x'))]_{P_i}\|_1 \geq 0\text{)} \\
&= \max_{x'}\big\|C_{\bar{c}}^1(\bar{\nabla}_S^1\mathcal{L}(x'))\big\|_1 = \bar{c}. \\
&\qquad \text{(by the definition of 1-norm)}
\end{aligned}$$

On the other hand, letting $\alpha = 2$ in Equation (7) derives

$$\begin{aligned}
&\Delta\mu_S^n \leq \max_{D,D'}\Big(\sum_{i=1}^{|S|}\|[C_{\bar{c}}^2(\bar{\mathbf{g}}_S^{\alpha}(x'))]_{P_i}\|_2^2\Big)^{\frac{1}{2}} \\
&= \max_{x'}\|C_{\bar{c}}^2(\bar{\mathbf{g}}_S^{\alpha}(x'))\|_2 = \bar{c}. \\
&\qquad \text{(by the definition of 2-norm)}
\end{aligned}$$

(2-b) The proof of $\bar{c} \leq \Delta\mu_S^n$: Since $\Delta\mu_S^n$ is the maximum value of $\|\mu_S^n(D) - \mu_S^n(D')\|_2$ for every pair of neighboring datasets D and D', we have $\Delta\mu_S^n = \max_{D,D'}\|\mu_S^n(D) - \mu_S^n(D')\|_2 \geq \|\mu_S^n(D'') - \mu_S^n(D''')\|_2$ for any pair of neighboring datasets D'' and D'''. Thus, let $D'' = \{\}$ and

$D''' = \{x'\}$ where x' is an example.

$$\begin{aligned}
\Delta\mu_S^n &= \max_{D,D'} \|\mu_S^n(D) - \mu_S^n(D')\|_2 \\
&\geq \|\mu_S^n(D'') - \mu_S^n(D''')\|_2 \\
&\geq \|\mu_S^n(\{\}) - \mu_S^n(\{x'\})\|_2 \\
&= \|\mu_S^n(\{x'\})\|_2 \qquad \text{(since } \mu_S^n(\{\}) = 0\text{)} \\
&= \Big(\sum_{i=1}^{|S|} (\mu_{S,P_i}^n(\{x'\}))^2\Big)^{1/2}. \quad \text{(by Definition 9)}
\end{aligned}$$

Since for any $P_i \in S$, we have

$$\begin{aligned}
&\Big(\sum_{i=1}^{|S|} (\mu_{S,P_i}^n(\{x'\}))^2\Big)^{1/2} \\
&\geq ((\mu_{S,P_i}^n(\{x'\}))^2)^{1/2} = \mu_{S,P_i}^n(\{x'\})
\end{aligned}$$

and $\mu_{S,P_i}^n(D) = \|[\sum_{x\in D} C_c^\alpha(\bar{\mathbf{g}}_S^\alpha(x))]_{P_i}\|_\alpha$ by Definition 8, we obtain the following

$$\Delta\mu_S^n \geq \mu_{S,P_1}^n(\{x'\}) = \|[C_{\bar{c}}^\alpha(\bar{\mathbf{g}}_S^\alpha(x'))]_{P_i}\|_\alpha.$$

Since the above inequality holds for any possible example x' and $0 \leq \|[C_c^\alpha(\bar{\mathbf{g}}_S^\alpha(x'))]_{P_1}\|_\alpha \leq \bar{c}$, we have $\bar{c} \leq \Delta\mu_S^n$.

Combining (1) $\Delta\mu_S^n \leq \bar{c}$ and (2) $\bar{c} \leq \Delta\mu_S^n$, we have

$$\Delta\mu_S^n = \bar{c} = c/(\frac{\sum_{P\in S}|P|}{|S|})^{1/\alpha}.$$

(3) The proof of $\Delta\mu_S^a = \bar{c} = c/(\frac{\sum_{P\in S}|P|}{|S|})^{1/\alpha}$: It is similar to the proof of $\Delta\mu_S^n = \bar{c} = c/(\frac{\sum_{P\in S}|P|}{|S|})^{1/\alpha}$. The only difference between μ_S^n and μ_S^a is whether we clip $\bar{\mathbf{g}}_S^\alpha(x)$ or $\mathrm{abs}(\bar{\mathbf{g}}_S^\alpha(x))$ for every example x. Furthermore, even though we apply any transformation to the gradient of an example x before clipping it with a clipping threshold $\bar{c}$, the α-norm of the clipped gradient of the example x is also bounded by $\bar{c}$ by Definition 3. Thus, in the proof of $\Delta\mu_S^n = \bar{c} = c/(\frac{\sum_{P\in S}|P|}{|S|})^{1/\alpha}$, if we replace $\bar{\mathbf{g}}_S^\alpha(x)$ and μ_S^n to $\mathrm{abs}(\bar{\mathbf{g}}_S^\alpha(x))$ and μ_S^a, respectively, we can prove that $\Delta\mu_S^a = \bar{c} = c/(\frac{\sum_{P\in S}|P|}{|S|})^{1/\alpha}$. □

Lemma 4. *Given $v_1, \ldots, v_M$, the log-likelihood $\ell(\tilde{v}_1, \ldots, \tilde{v}_t)$ of noisy PGMs is maximized by $\lambda_j = (\sum_{P_i\in R_{j-1}} \frac{\tilde{v}_{j,i}v_i}{\Delta_j^2})/(\sum_{P_i\in R_{j-1}} \frac{v_i^2}{\Delta_j^2})$ with $1 \leq j \leq t$.*

Proof. Recall that the log-likelihood $\ell(\tilde{v}_1, \ldots, \tilde{v}_t)$ is

$$\begin{aligned}
&\ell(\tilde{v}_1, \ldots, \tilde{v}_t) = \\
&-\sum_{t'=1}^{t} \sum_{P_i\in R_{t'-1}} \log(\sigma_{\mathrm{ps}}\Delta_{t'}\sqrt{2\pi}) \\
&-\sum_{t'=1}^{t} \frac{1}{2(\sigma_{\mathrm{ps}}\Delta_{t'})^2} \sum_{P_i\in R_{t'-1}} (\tilde{v}_{t',i} - \lambda_{t'}v_i)^2.
\end{aligned} \tag{8}$$

In Equation (8), since σ_{ps}, Δ_1, Δ_2, …, Δ_{t-1} and Δ_t are constants, $-\sum_{t'=1}^{t}\sum_{P_i\in R_{t'-1}} \log(\sigma_{\mathrm{ps}}\Delta_{t'}\sqrt{2\pi})$ is constant. Thus, maximizing the log-likelihood $\ell(\tilde{v}_1, \ldots, \tilde{v}_t)$ is equivalent to maximizing the following objective function:

$$\begin{aligned}
&\ell_o(\tilde{v}_1, \ldots, \tilde{v}_t) \\
&= -\sum_{t'=1}^{t} \sum_{P_i\in R_{t'-1}} \frac{\left(\tilde{v}_{t',i} - \lambda_{t'}v_i\right)^2}{\Delta_{t'}^2}
\end{aligned} \tag{9}$$

Furthermore, we have

$$\frac{\partial\ell_o(\tilde{v}_1, \ldots, \tilde{v}_t)}{\partial\lambda_{t'}} = -2 \sum_{P_i\in R_{t'-1}} \frac{v_i}{\Delta_{t'}^2}\left(\tilde{v}_{t',i} - \lambda_{t'}v_i\right).$$

By setting the derivative $\frac{\partial\ell_o(\tilde{v}_1,\ldots,\tilde{v}_t)}{\partial\lambda_{t'}}$ to zero, we have

$$\sum_{P_i\in R_{t'-1}} \frac{v_i}{\Delta_{t'}^2}\tilde{v}_{t',i} = \sum_{P_i\in R_{t'-1}} \lambda_{t'}\frac{v_i^2}{\Delta_{t'}^2}.$$

By re-arranging the above equation, we get

$$\lambda_{t'} = \frac{\sum_{P_i\in R_{t'-1}} \frac{\tilde{v}_{t',i}v_i}{\Delta_{t'}^2}}{\sum_{P_i\in R_{t'-1}} \frac{v_i^2}{\Delta_{t'}^2}}.$$ □

Lemma 5. *Given $\lambda_1 = q_{\mathrm{ps}}$, $\lambda_2, \ldots, \lambda_t$, the log-likelihood $\ell(\tilde{v}_1, \ldots, \tilde{v}_t)$ of noisy PGMs is maximized by $v_i = (\sum_{j\in\mathcal{T}_{t,i}} \frac{\tilde{v}_{j,i}\lambda_j}{\Delta_j^2})/(\sum_{j\in\mathcal{T}_{t,i}} \frac{\lambda_j^2}{\Delta_j^2})$ where $\mathcal{T}_{t,i} = \{j \mid P_i \in R_{j-1} \wedge 1 \leq j \leq t\}$ with $1 \leq i \leq M$.*

Proof. In Equation (8), since σ_{ps}, Δ_1, Δ_2, …, Δ_{t-1} and Δ_t are constants, $-\sum_{t'=1}^{t}\sum_{P_i\in R_{t'-1}} \log(\sigma_{\mathrm{ps}}\Delta_{t'}\sqrt{2\pi})$ is constant. Thus, maximizing the log-likelihood $\ell(\tilde{v}_1, \ldots, \tilde{v}_t)$ is equivalent to maximizing the objective function $\ell_o(\tilde{v}_1, \ldots, \tilde{v}_t)$ in Equation (9). Furthermore, we have

$$\frac{\partial\ell_o(\tilde{v}_1, \ldots, \tilde{v}_t)}{\partial v_i} = -2 \sum_{t'\in\mathcal{T}_{t,i}} \frac{\lambda_{t'}}{\Delta_{t'}^2}\left(\tilde{v}_{t',i} - \lambda_{t'}v_i\right).$$

By setting the derivative $\frac{\partial \ell_o(\tilde{v}_1,\ldots,\tilde{v}_t)}{\partial v_i}$ to zero, we have

$$\sum_{t' \in \mathcal{T}_{t,i}} \frac{\lambda_{t'}}{\Delta_{t'}^2} \tilde{v}_{t',i} = \sum_{t' \in \mathcal{T}_{t,i}} \frac{\lambda_{t'}^2}{\Delta_{t'}^2} v_i.$$

By re-arranging the above equation, we get

$$v_i = \frac{\sum_{t' \in \mathcal{T}_{t,i}} \frac{\tilde{v}_{t',i}\lambda_{t'}}{\Delta_{t'}^2}}{\sum_{t' \in \mathcal{T}_{t,i}} \frac{\lambda_{t'}^2}{\Delta_{t'}^2}}. \qquad \square$$

Lemma 6. *Given $\lambda_1, \ldots, \lambda_t$, the variance of v_i, denoted by $\mathrm{Var}(v_i)$, is $\sigma_{\mathrm{ps}}^2/(\sum_{j=1}^t \frac{\lambda_j^2}{\Delta_j^2})$ for every partition $P_i \in R_t$ at the t-th iteration.*

Proof. For $P_i \in R_{t-1}$, by Lemma 5, we have

$$\mathrm{Var}(v_i) = \mathrm{Var}\left(\frac{\sum_{t'=1}^t \frac{\tilde{v}_{t',i}\lambda_{t'}}{\Delta_{t'}^2}}{\sum_{t'=1}^t \frac{\lambda_{t'}^2}{\Delta_{t'}^2}} \right).$$

For a random variable v, let $\mathrm{Var}(v)$ be the variance of v. Then, we have

$$\mathrm{Var}(av) = a^2 \,\mathrm{Var}(v)$$

where a is a constant (Cormen et al., 2022). Since $\lambda_{t'}$s and $\Delta_{t'}$s are constant, for $P_i \in R_{t-1}$, we get

$$\mathrm{Var}(v_i) = \frac{\sum_{t'=1}^t \frac{\lambda_{t'}^2}{\Delta_{t'}^4} \mathrm{Var}(\tilde{v}_{t',i})}{\left(\sum_{t'=1}^t \frac{\lambda_{t'}^2}{\Delta_{t'}^2}\right)^2}.$$

Since $\tilde{v}_{t,i}$ has the following probability density function f of the Gaussian distribution

$$f(\tilde{v}_{t,i}) = \frac{1}{\sigma_{\mathrm{ps}}\Delta_t\sqrt{2\pi}} \exp\Big(-\frac{1}{2}\Big(\frac{\tilde{v}_{t,i} - \lambda_t v_i}{\sigma_{\mathrm{ps}}\Delta_t}\Big)^2\Big),$$

the variance of $\tilde{v}_{t,i}$ (i.e., $\mathrm{Var}(\tilde{v}_{t,i})$) is $\sigma_{\mathrm{ps}}^2\Delta_{t'}^2$. Thus, for $P_i \in R_{t-1}$, we obtain

$$\mathrm{Var}(v_i) = \frac{\sigma_{\mathrm{ps}}^2\left(\sum_{t'=1}^t \frac{\lambda_{t'}^2}{\Delta_{t'}^2}\right)}{\left(\sum_{t'=1}^t \frac{\lambda_{t'}^2}{\Delta_{t'}^2}\right)^2} = \frac{\sigma_{\mathrm{ps}}^2}{\sum_{t'=1}^t \frac{\lambda_{t'}^2}{\Delta_{t'}^2}}. \qquad \square$$

B Computing the Noise Multiplier

We present the *ComputeSigma* algorithm that computes the noise multiplier σ_{SGD} for the DP-SGD algorithm and the noise multiplier σ_{SGD} for the *PSEL* algorithm. It first computes the minimum noise multiplier σ_{SGD} for the DP-SGD algorithm such that the consumed privacy budget by the DP-SGD algorithm does not exceed $r_\epsilon \cdot \epsilon$ where r_ϵ is the privacy budget ratio for the DP-SGD algorithm. By using the obtained noise multiplier σ_{SGD}, we next find the minimum noise multiplier σ_{ps}, which will be used by the *PSEL* algorithm, such that the total privacy budget used by the DP-SGD algorithm and the *PSEL* algorithm does not exceed ϵ. To compute the minimum noise multipliers σ_{SGD} or σ_{ps}, we utilize the library *opacus*[2], which provides the procedures to compute the privacy budget consumption ϵ by the composition of algorithms based on the RDP accountant method in (Mironov et al., 2019; Balle et al., 2020).

Applying RDP accountant method: For an example set D and a set of algorithms $A_1, A_2, \ldots, A_L$, assume that each algorithm A_i performs the following noise insertion procedure T_i times.

1. Obtain a set X of examples by sampling each example in D with probability q_i.
2. Insert noise sampled from $\mathcal{N}(0, \Delta^2\sigma_i^2\mathbf{I})$ to the output of a function with the sensitivity Δ where the function takes the set X of examples as input.

Since the spent privacy budget by each algorithm A_i is determined by σ_i, q_i and T_i, let us denote the account representation of the algorithm A_i by (σ_i, q_i, T_i). Furthermore, let us denote the composition of the algorithms A_is by an account representation set $S_{\mathrm{AR}} = \{(\sigma_1, q_1, T_1), (\sigma_2, q_2, T_2), \ldots, (\sigma_L, q_L, T_L)\}$. Given a failure constant δ and an account representation set S_{AR}, we invoke the *RDPAccountant* function provided in the *opacus* library to compute the total privacy budget consumption ϵ' based on the RDP accountant method.

Finding the noise multiplier σ given a privacy budget ϵ: Given the noise multipliers $\sigma_1, \sigma_2, \ldots, \sigma_L$ of algorithms $A_1, A_2, \ldots, A_L$, we want to find the minimum noise multiplier σ_{L+1} of an algorithm A_{L+1} such that the total privacy budget ϵ' consumed by $A_1, A_2, \ldots, A_{L+1}$ does not exceed a given privacy budget ϵ. Since the total privacy budget consumption ϵ' computed by the *RDPAccountant* function decreases as σ_{L+1} increases, we find the minimum noise multiplier σ_{L+1} with the binary search by calling the *ComputeNoiseMultiplier* procedure.

Its pseudocode is provided in Algorithm 2. We first compute an upper bound of the minimum σ_{L+1} by repeatedly doubling σ_{high} from 10 until the consumed privacy budget ϵ_{high} becomes not larger than

[2] https://github.com/pytorch/opacus

Algorithm 2 *ComputeNoiseMultiplier*

Input: A privacy budget ϵ, a failure constant δ, an account representation set $S_{\mathrm{AR}} = \{(\sigma_1, q_1, T_1), (\sigma_2, q_2, T_2), \ldots, (\sigma_L, q_L, T_L)\}$, a sampling rate q_{L+1}, the number of steps T_{L+1} and a budget tolerance τ

1: $\epsilon_{\mathrm{high}} \leftarrow \infty$
2: $\sigma_{\mathrm{high}} \leftarrow 10$
3: **while** $\epsilon_{\mathrm{high}} > \epsilon$ **do**
4: $\quad \sigma_{\mathrm{high}} \leftarrow 2 \cdot \sigma_{\mathrm{high}}$
5: $\quad \epsilon_{\mathrm{high}} \leftarrow$ *RDPAccountant*$(\delta, S_{\mathrm{AR}} \cup \{(\sigma_{\mathrm{high}}, q_{L+1}, T_{L+1})\})$
6: $\sigma_{\mathrm{low}} \leftarrow 0$
7: **while** $(\epsilon - \epsilon_{\mathrm{high}}) > \tau$ **do**
8: $\quad \sigma' \leftarrow \frac{\sigma_{\mathrm{low}} + \sigma_{\mathrm{high}}}{2}$
9: $\quad \epsilon' \leftarrow$ *RDPAccountant*$(\delta, S_{\mathrm{AR}} \cup \{(\sigma', q_{L+1}, T_{L+1})\})$
10: $\quad$ **if** $\epsilon' < \epsilon$ **then**
11: $\quad\quad \sigma_{\mathrm{high}} \leftarrow \sigma'$
12: $\quad\quad \epsilon_{\mathrm{high}} \leftarrow \epsilon'$
13: $\quad$ **else**
14: $\quad\quad \sigma_{\mathrm{low}} \leftarrow \sigma'$
15: **return** σ'

ϵ (lines 1-5). Then, we perform the binary search by updating the range of $[\sigma_{\mathrm{low}}, \sigma_{\mathrm{high}}]$ until the difference between ϵ and ϵ' less than τ (lines 8-14). Specifically, it first computes the budget consumption ϵ' for $\sigma' = (\sigma_{\mathrm{low}} + \sigma_{\mathrm{high}})/2$. We update the range to $[\sigma_{\mathrm{low}}, \sigma']$ if $\epsilon' < \epsilon$ and $[\sigma', \sigma_{\mathrm{high}}]$ otherwise. This process is repeated until $\epsilon - \epsilon_{\mathrm{high}} \leq \tau$ to find the minimum σ_{L+1} with satisfying (ϵ, δ)-differential privacy.

The *ComputeSigma* algorithm: We present the *ComputeSigma* algorithm that computes the noise multiplier σ_{SGD} for the DP-SGD algorithm and the noise multiplier σ_{SGD} for the *PSEL* algorithm. Its pseudocode is presented in Algorithm 3. We first set q_{SGD} and T_{SGD} to the sampling rate and the number of iterations used by the DP-SGD algorithm (lines 1-2). Since we do not consider any other algorithm than the DP-SGD algorithm during computation of σ_{SGD}, we set the account representation set S_{AR} to an empty set (line 3). Then, we set σ_{SGD} to the noise multiplier computed by invoking the *ComputeNoiseMultiplier* with the privacy budget $r_\epsilon \cdot \epsilon$, the account representation set S_{AR}, the sampling rate q_{SGD} and the number of iterations T_{SGD} (line 4). Since we also consider the DP-SGD algorithm with the determined noise multiplier σ_{SGD} while computing the noise multiplier σ_{ps} for the *PSEL* algorithm, we set the account representation set S_{AR} to the set only containing the account representation $(\sigma_{\mathrm{SGD}}, q_{\mathrm{SGD}}, T_{\mathrm{SGD}})$ of the DP-SGD algorithm (line 5). Then, we set σ_{ps} to the noise multiplier computed by invoking the *ComputeNoiseMultiplier* with the privacy budget ϵ, the account representation set S_{AR}, the sampling rate q_{ps} and the number of iterations T_{ps} (line 6). Finally, the *ComputeSigma* algorithm returns the noise multipliers σ_{SGD} and σ_{ps} (line 7).

Algorithm 3 *ComputeSigma*

Input: A dataset D, a privacy budget ϵ, a failure constant δ, a batch size B, the number of epochs E, a budget ratio for DP-SGD r_ϵ, a budget tolerance τ, the number of iterations during parameter selection T_{ps}, the sampling rate of parameter selection q_{ps}

1: $q_{\mathrm{SGD}} \leftarrow B/|D|$
2: $T_{\mathrm{SGD}} \leftarrow \lfloor E \cdot |D|/B \rfloor$
3: $S_{\mathrm{AR}} \leftarrow \{\}$
4: $\sigma_{\mathrm{SGD}} \leftarrow$ *ComputeNoiseMultiplier*$(r_\epsilon\epsilon, \delta, S_{\mathrm{AR}}, q_{\mathrm{SGD}}, T_{\mathrm{SGD}}, \tau)$
5: $S_{\mathrm{AR}} \leftarrow \{(\sigma_{\mathrm{SGD}}, q_{\mathrm{SGD}}, T_{\mathrm{SGD}})\}$
6: $\sigma_{\mathrm{ps}} \leftarrow$ *ComputeNoiseMultiplier*$(\epsilon, \delta, S_{\mathrm{AR}}, 1, T_{\mathrm{ps}}, \tau)$
7: **return** $\sigma_{\mathrm{SGD}}, \sigma_{\mathrm{ps}}$

C The Details of the Proposed Algorithms

The DP-FROST algorithm in Section 4.7 computes the input noise multipliers σ_{ps} of the *PSEL* algorithm and σ_{SGD} of the DP-SGD algorithm by invoking the *ComputeSigma* algorithm. Furthermore, it selects the parameters to be fine-tuned by invoking the *PSEL* algorithm. When the *PSEL* algorithm selects the partitions to be fine-tuned, it utilizes estimated $\mu^a_{S,P_i}(D)$s of partitions by invoking the *PGM-EST* algorithm. We provide the pseudocode and the details of the *ComputeSigma*, *PSEL* and *PGM-EST* algorithms.

C.1 The *PSEL* Algorithm

Its pseudocode is provided in Algorithm 4. Let S be the partition set $\{P_1, P_2, \ldots, P_M\}$ of the disjoint partitions of Θ. We first set the remaining partition set R_0 to the partition set S and set the selected partition set S_1 to $\{\}$ (line 1). Then, for each t-th iteration with $t = 1, \ldots, T$, we repeatedly perform the following steps.

1. Obtain a set of examples X_t by sampling each

Algorithm 4 *PSEL*

Input: A dataset D, a partition set S, a noise multiplier σ_{ps}, a norm order α, the number of maximum iterations T, a unfreezing ratio γ, a gap coefficient ν, the sampling rate of parameter selection q_{ps}

1: $R_0 \leftarrow S$, $S_1 \leftarrow \{\}$
2: **for** $t \leftarrow 1$ **to** T **do**
3: $X_t \leftarrow$ a subset of D by sampling each example with probability q_{ps}
4: $\Delta_t \leftarrow \Delta\mu^a_{R_{t-1}}$ by Lemma 3
5: **for** $P_i \in R_{t-1}$ **do**
6: Compute $\mu^a_{R_{t-1},P_i}(X_t)$ by Definition 8
7: Sample z from $\mathcal{N}(0, \sigma^2_{\text{ps}}\Delta^2_t)$
8: $\tilde{v}_{t,i} \leftarrow \mu^a_{R_{t-1},P_i}(X_t) + z$
9: $v_1, \ldots, v_M \leftarrow$ *PGM-EST*$(\tilde{v}_1, \ldots, \tilde{v}_t, \Delta_1, \ldots, \Delta_t)$
10: $\sigma_{v_i} \leftarrow \sqrt{\frac{\sigma^2_{\text{ps}}\Delta^2_{t'}}{1+\sum_{t'=1}^{t}\lambda^2_{t'}}}$ by Lemma 6
11: Sort partitions P_is in R_{t-1} with non-decreasing order of v_i
12: Assume that $v_{i_1} \le v_{i_2} \le \cdots \le v_{i_{|R_{t-1}|}}$
13: $j \leftarrow 1$
14: $n_p \leftarrow |P_{i_j}|$
15: **while** $n_p < (1-\gamma)|\Theta|$ **do**
16: $j \leftarrow j + 1$
17: $n_p \leftarrow n_p + |P_{i_j}|$
18: $\mu_{\text{thres}} \leftarrow v_{i_j}$
19: $j \leftarrow |R_{t-1}|$
20: **while** $|P_{i_j}| + \sum_{P\in S_t}|P| \le t\gamma|\Theta|/T$ **do**
21: **if** $t = T$ **or** $v_{i_j} > \mu_{\text{thres}} + \nu\sigma_{v_{i_j}}$ **then**
22: $S_t \leftarrow S_t \cup \{P_{i_j}\}$
23: $j \leftarrow j - 1$
24: $R_t \leftarrow R_{t-1} - S_t$
25: **return** $\bigcup_{P\in S_T} P$

example in D with probability q_{ps} (line 3).

2. For each partition $P_i \in R_{t-1}$, compute $\tilde{v}_{t,i}$ by Equation (1) (lines 4-8).

3. For $i = 1, \ldots, M$, set v_i to the estimated true $\mu^a_{R_{t-1},P_i}(D)$ by the *PGM-EST* algorithm (line 9).

4. Set σ_{v_i} to the standard deviation of v_i for $P_i \in R_{t-1}$ by Lemma 6 (line 10).

5. Sort partitions P_is in R_{t-1} with non-decreasing order of v_i. Assume that $v_{i_1} \le v_{i_2} \le \cdots \le v_{i_{|R_{t-1}|}}$ (lines 11-12).

6. Set the PGM threshold μ_{thres} to v_j where $j = \text{argmin}_{j'} \sum_{k=1}^{j'} |P_k| \ge (1-\gamma)|\Theta|$ (lines 13-18).

7. When $t < T$, add additional partitions P_is with the largest v_is satisfying $v_i > \mu_{\text{thres}} + \nu\sigma_{v_i}$ in R_{t-1} to the selected partition set S_t as long as $\sum_{P_i\in S_t}|P_i| \le t\cdot\gamma\cdot|\Theta|/T$. When $t = T$, we select the partitions P_is with the largest v_is such that the total number of parameters in the selected partitions does not exceed $\gamma\cdot|\Theta|$ (lines 19-23).

8. Set the remaining partition set R_t to the set obtained by excluding S_t from R_{t-1} (line 24).

After finishing the outer for-loop in lines 2-24, we return the set of selected parameters in S_T (line 25).

C.2 The *PGM-EST* Algorithm

Its pseudocode is presented in Algorithm 5. If $t = 1$ (i.e., the first iteration), we immediately return the noisy $\mu^a_{S,P_i}(D)$s of all partitions (lines 1-2). If $t \ge 2$, we first initialize $\lambda_1, \ldots, \lambda_t$ to 1 (lines 3-4). Then, we repeatedly perform the following steps J times: (1) calculate $v_1, \ldots, v_M$ to maximize $\ell_o(\tilde{v}_1, \ldots, \tilde{v}_t)$ with the current values of $\lambda_1, \ldots, \lambda_t$ by Lemma 5 (line 6-7), and (2) update $\lambda_1, \ldots, \lambda_t$ to maximize $\ell_o(\tilde{v}_1, \ldots, \tilde{v}_t)$ with the current values of $v_1, \ldots, v_M$ by Lemma 4 (line 8-9). After finishing the outer for-loop in lines 5-9, we return $v_1, \ldots, v_M$.

D Additional Experimental Study

D.1 Ablation Study

We compare the accuracy of fine-tuned models with different parameter selection methods on the SST-2,

Method	α	SST-2	QNLI	QQP	MNLI
ℓ_1-DP-MG (SINGLE)	1	87.96	81.84	79.81	74.27
ℓ_1-DP-MG	1	88.11	82.18	80.85	74.49
ℓ_1-DP-MGN (SINGLE)	1	87.54	82.52	82.44	75.26
ℓ_1-DP-MGN	1	88.65	82.36	**83.83**	79.77
ℓ_1-DP-MGNA (SINGLE)	1	89.56	83.64	**83.83**	**80.35**
ℓ_1-DP-MGNA	1	**90.71**	**84.14**	83.76	79.86
ℓ_2-DP-MG (SINGLE)	2	88.26	81.84	80.43	74.35
ℓ_2-DP-MG	2	88.34	82.22	81.35	74.71
ℓ_2-DP-MGN (SINGLE)	2	90.10	82.06	83.23	77.58
ℓ_2-DP-MGN	2	89.60	83.16	83.16	77.57
ℓ_2-DP-MGNA (SINGLE)	2	90.41	83.60	83.15	77.40
ℓ_2-DP-MGNA	2	90.02	83.70	82.94	77.27

Table 7: Comparison of parameter selection methods

Algorithm 5 *PGM-EST*

Input: $\tilde{v}_1, \ldots, \tilde{v}_t, \Delta_1, \ldots, \Delta_t$, the number of maximum iterations J, the number of partitions M

if $t = 1$ **then**
  return $\tilde{v}_{1,1}, \ldots, \tilde{v}_{1,M}$
for $t' \leftarrow 2$ **to** t **do**
  $\lambda_{t'} \leftarrow 1$
for $j \leftarrow 1$ **to** J **do**
  for $i \leftarrow 1$ **to** M **do**
    $v_i \leftarrow \frac{\sum_{t' \in \mathcal{T}_{t,i}} \frac{\tilde{v}_{t',i} \lambda_{t'}}{\Delta^2_{t'}}}{\sum_{t' \in \mathcal{T}_{t,i}} \frac{\lambda^2_{t'}}{\Delta^2_{t'}}}$ by Lemma 5
  for $t' \leftarrow 2$ **to** t **do**
    $\lambda_{t'} \leftarrow \frac{\sum_{P_i \in R_{t'-1}} \frac{\tilde{v}_{t',i} v_i}{\Delta^2_{t'}}}{\sum_{P_i \in R_{t'-1}} \frac{v_i^2}{\Delta^2_{t'}}}$ by Lemma 4
return $v_1, \ldots, v_M$

QNLI, QQP and MNLI for $\epsilon = 0.5$. The results are shown in Tables 7. Note that inserted noise to satisfy the (ϵ, δ)-differential privacy is stronger when $\epsilon = 0.5$ than when $\epsilon > 0.5$. DP-MG, DP-MGN and DP-MGNA denote the DP-FROST methods with the corresponding private PGM computation methods, respectively. Moreover, SINGLE indicates the variant of DP-FROST that selects partitions at once without using the *PSEL* algorithm. DP-MGN outperforms DP-MG in most cases and it indicates the effectiveness of DP-MGN by reducing the sensitivity of partition-wise PGMs. Moreover, DP-MGNA with $\alpha = 1$ tends to show stable performance such that the difference between the accuracies of DP-MGNA and the second-best performer is at most 0.07 %. Thus, it is desirable to employ DP-MGNA with $\alpha = 1$ to achieve the accuracy close to the best performer.

Table 8: The accuracy of text classification using RoBERTa-large

Method	SST-2	QNLI	QQP	MNLI
Full	89.98	86.78	84.68	84.57
PVAL	90.44	86.61	84.86	84.66
LoRA	91.97	86.84	**84.91**	85.19
Adapter	90.18	84.70	84.16	85.09
DP-FROST	**92.24**	**87.74**	84.86	**85.47**

Table 9: The training time (seconds) using RoBERTa-large

Method	SST-2	QNLI	QQP	MNLI
Full	5036	24618	240914	305815
PVAL	5904	28219	266399	400021
LoRA	1466	9375	65523	97821
Adapter	1221	8871	64273	92226
DP-FROST	1625	9763	66969	99214

D.2 Experiments with RoBERTa-large

While we compared our methods with the state-of-the-art for differentially private fine-tuning methods by using the pre-trained RoBERTa-Base (Liu et al., 2019) in Section 5, we also show the text classification accuracies with RoBERTa-large on SST-2, QNLI, QQP and MNLI for $\epsilon = 0.5$ in Table 8. Similar to the results with RoBERTa-Base, our DP-FROST method outperforms the other methods on SST-2, QNLI and MNLI. On QQP, the DP-FROST method achieves second-best performance and the gap between the accuracies of DP-FROST and the second-best performer is at most 0.05 %. Thus, we can conclude that DP-FROST is effective for large models too.

Table 9 shows the training times using RoBERTa-large on SST-2, QNLI, QQP and MNLI for $\epsilon = 0.5$. Except that it takes longer time to train RoBERTa-large than RoBERTa-Base, the results using RoBERTa-large in Table 9 show similar

trends to the results using RoBERTa-Base in Table 5. On MNLI dataset that takes the longest training time among all the tested datasets, applying our method to RoBERTa-large increases the training time by only 7.6 % compared to applying the fastest Adapter method.