SAMPLE-EFFICIENT DIFFERENTIALLY PRIVATE FINE-TUNING VIA GRADIENT MATRIX DENOISING

Anonymous authors
Paper under double-blind review

ABSTRACT

We address the challenge of sample efficiency in differentially private fine-tuning of large language models (LLMs) using DP-SGD. While DP-SGD provides strong privacy guarantees, the added noise significantly increases the entropy of gradient matrices, disrupting their low-rank structure and slowing optimization. We propose a post-processing algorithm that leverages random matrix theory to denoise gradients, restore low-rank structure, and improve alignment with the original signal. Applied to DP-SGD fine-tuning of RoBERTa on GLUE tasks, our method improves sample efficiency compared to state-of-the-art approaches, substantially reducing training time when optimal performance is not required. This work demonstrates that matrix recovery techniques can enhance the utility of private language model training without compromising privacy guarantees.

1 INTRODUCTION

Many applications of machine learning in natural language processing tasks may raise privacy concerns, because of the potential data leakage from using models trained on private data (Carlini et al., 2021; 2022). Differential privacy (DP) (Dwork et al., 2014) is a formal framework for quantifying and limiting the privacy loss experienced by individuals whose data are included in a dataset when an algorithm is applied to it. DP-SGD (Abadi et al., 2016), is a method to ensure privacy guarantees as measured by the DP framework, and has been succesfully applied to NLP tasks (Yu et al., 2021; Li et al., 2021).

Applying DP-SGD to language models, while successful, has many challenges. Training large language models (LLMs) with DP-SGD is computationally expensive (Li et al., 2021). Using parameter efficient fine-tuning methods, this challenge has been addressed (Yu et al., 2021). Still, computational cost is higher than the non-private training, because of lower sample efficiency.

In the DP-SGD method, noise is deliberately added to the gradient vector before it is passed to the optimizer to ensure privacy. While this step is crucial for protecting individual data, it also complicates the optimization process. Specifically, the added noise alters the distribution of singular values in the gradient matrix. For transformer-based language models, the singular values of the gradient matrix typically decay rapidly, reflecting low matrix entropy and a strong low-rank structure (Li et al., 2022; Zhao et al., 2024). After noise is introduced, however, the singular values decay more slowly, leading to higher matrix entropy (Li et al., 2022). We hypothesize that this increase in entropy makes optimization more difficult.

The singular values of the gradient matrix undergo a "phase transition" (Baik et al., 2005) when noise is added. If the underlying signal is weak, the singular values of the noisy matrix become indistinguishable from those of pure noise. Figure 1 illustrates this by comparing the sorted singular values of a RoBERTa layer's gradient matrix before and after DP-SGD noise is applied. In this weak-signal regime, the noisy gradient's singular values closely follow the "bulk" distribution predicted by the Marchenko–Pastur law (Marčenko & Pastur, 1967; Tao, 2012), making them essentially indistinguishable from pure noise. Thus, when a low-rank signal is too small relative to the noise, it is hidden in the noise and cannot be detected or recovered by examining the singular values and vectors alone. This highlights a fundamental limitation: sufficiently weak signals are undetectable in the presence of strong noise.

However, if some singular values exceed this threshold, the largest singular values of the noisy matrix deviate from the bulk, as shown in Figure 2. This phenomenon is known as the Baik–Ben Arous–Péché (BBP) phase transition (Baik et al., 2005). The extent of these deviations, as well as the alignment between the singular vectors of the noisy and original matrices, can be predicted mathematically (Baik & Silverstein, 2006; Benaych-Georges & Nadakuditi, 2012). These properties enable partial recovery of the original matrix from its noisy observation (Shabalin & Nobel, 2013; Gavish & Donoho, 2014).

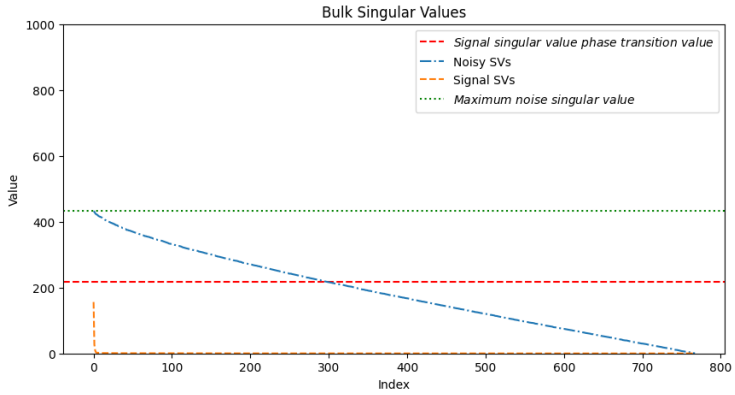

Figure 1: Sorted singular values of the gradient matrix for a RoBERTa layer, before and after adding DP-SGD noise. When the signal singular values are smaller than the red line, the singular values of the noisy matrix are indistinguishable from pure noise.

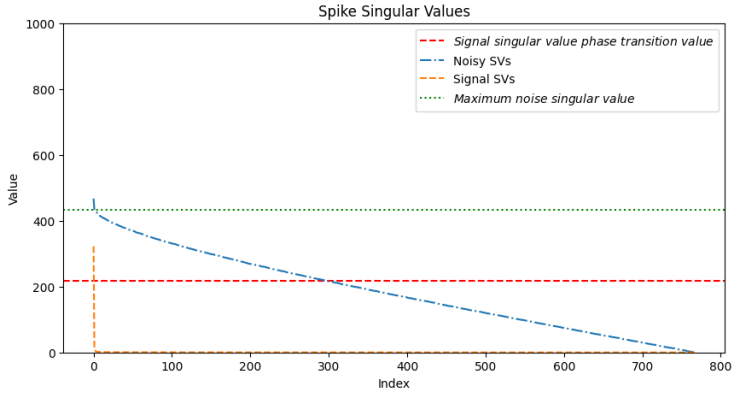

Figure 2: Sorted singular values of the gradient matrix for a RoBERTa layer, before and after adding DP-SGD noise. When some signal singular values exceed the red line, the largest singular values of the noisy matrix deviate from the bulk.

In this paper, we propose a post-processing algorithm for DP-SGD that leverages the mentioned matrix recovery techniques from random matrix theory to reduce the entropy of the gradient matrix, restore its low-rank structure, and improve the alignment between the noisy and original gradients. To evaluate our approach, we apply it to DP-SGD fine-tuning of RoBERTa (Liu et al., 2019) on GLUE tasks (Wang et al., 2019). We compare the sample efficiency of our method to the current state-of-the-art (Yu et al., 2021), demonstrating that our approach can improve the sample efficiency of DP-SGD fine-tuning for language models. While our method may not always achieve the highest possible utility, it can substantially reduce training time when optimal performance is not required.

2 PRELIMINARIES

2.1 DIFFERENTIAL PRIVACY

Differential privacy is a framework to quantify and measure the maximum possible privacy risks an algorithm with sensitive training dataset may have. For a pair (ϵ, δ), this formalism asks any learning algorithm $\mathcal{M}$ to have similar outputs for two datasets differing only in one element. Intuitively, the output of the learning algorithm should not change much whether it sees a particular example or not. This intuition can be formulated mathematically in the concept of approximate differential privacy.

2.1.1 APPROXIMATE DIFFERENTIAL PRIVACY

Definition 1. *Two sets are called neighboring sets if they differ only in inclusion or exclusion of exactly one element.*

Definition 2. *A randomized algorithm $\mathcal{M}$ is said to satisfy (ϵ, δ) differential privacy, if for any two neighboring datasets D and D' and for any event E, the following holds*

$$\mathbb{P}(\mathcal{M}(D) \in E) \leq \exp(\epsilon)\mathbb{P}(\mathcal{M}(D') \in E) + \delta \tag{1}$$

.

In practice, it is usual to have δ in the order of $|D|^{-1}$ (Abadi et al., 2016). In NLP applications, ϵ usually takes values between 0.5 and 8 (Yu et al., 2021; Li et al., 2021).

2.1.2 DP-SGD

DP-SGD is a popular method of training deep learning models with approximate differential privacy guarantees. This method is a modification of the popular first order SGD algorithm.

DP-SGD works by modifying the gradient before passing it to the optimizer. It has two main parts 1. per example gradient clipping and 2. noise addition. There are two hyper-parameters associated with each of them, the clipping threshold, C, which controls the maximum per example gradient norm, and the noise multiplier, σ, which when multiplied by C, controls the standard deviation of the isotropic zero mean Gaussian noise added to the sum of the clipped gradients (Abadi et al., 2016).

In standard SGD, for a batch of data $\{x_i\}_{i\in\mathcal{B}} \subset D$, the batch gradient is computed as:

$$\boldsymbol{g}_{\mathcal{B}} = \frac{1}{B}\sum_{i\in\mathcal{B}} \boldsymbol{g}_i = \frac{1}{B}\sum_{i\in\mathcal{B}} \nabla f(\theta, x_i)$$

In DP-SGD, each individual gradient is first clipped so that its norm does not exceed the threshold C. The clipped gradients are then summed, and Gaussian noise with entries drawn from $\mathcal{N}(0, \sigma C)$ is added. Finally, the result is averaged over the batch:

$$\begin{aligned} \bar{\boldsymbol{g}}_{\mathcal{B}} &= \sum_{i\in\mathcal{B}} \text{clip}(\boldsymbol{g}_i, C) \\ \tilde{\boldsymbol{g}}_{\mathcal{B}} &= \frac{1}{B}\left(\bar{\boldsymbol{g}}_{\mathcal{B}} + \boldsymbol{w}\right), \quad \boldsymbol{w}_j \sim \mathcal{N}(0, \sigma C) \end{aligned} \tag{2}$$

This new gradient will then be fed to the optimizing algorithm of the choice, e.g. SGD or Adam(W). While σ and C are hyper-parameters, the constant σ is selected based on the privacy guarantees desired for the model (ϵ, δ), the number of training steps, sampling rate $(\frac{B}{|D|})$. The method for computing the necassiry σ based on the privacy gaurantees is called the privacy accountant. For this work, we use the privacy accountant of Gopi et al. (2021) which currently is the most tight privacy accountant.

One other important aspect of DP-SGD is the sampling mechanism. Most of the privacy accountants rely on the notion of "sampling with replacement" to ensure that each data point is independently and identically distributed (i.i.d.) during training. So the sampling process must account for this to maintain the desired privacy guarantees.

Algorithm 1 DP-SGD

Require: Dataset D, loss function $f(\theta, x)$, model parameters θ, sampling rate ρ, clipping norm C, noise multiplier σ, optimizer $\mathcal{O}$, number of steps T
  for $t \leftarrow 1$ to T **do**
    Sample a batch $\mathcal{B}$ from D using Poisson sampling with rate ρ
    for each $i \in \mathcal{B}$ **do**
      Compute per-example gradient $\boldsymbol{g}_i \leftarrow \nabla_\theta f(\theta, x_i)$
      Clip gradient: $\mathrm{clip}(\boldsymbol{g}_i, C) \leftarrow \boldsymbol{g}_i / \max(1, \|\boldsymbol{g}_i\|_2 / C)$
    end for
    Aggregate clipped gradients: $\bar{\boldsymbol{g}} \leftarrow \sum_{i \in \mathcal{B}} \mathrm{clip}(\boldsymbol{g}_i, C)$
    Draw noise vector $\boldsymbol{w}$ with i.i.d. entries from $\mathcal{N}(0, \sigma^2 C^2)$
    Compute noisy average: $\tilde{\boldsymbol{g}} \leftarrow (\bar{\boldsymbol{g}} + \boldsymbol{w})/|\mathcal{B}|$
    Update optimizer state: $\mathcal{O} \leftarrow \mathrm{UpdateState}(\mathcal{O}, \tilde{\boldsymbol{g}})$
    Update parameters: $\theta \leftarrow \mathrm{UpdateParameters}(\theta, \mathcal{O})$
  end for

2.1.3 POST PROCESSING INVARIANCE

A fundamental property of differential privacy is its invariance under post-processing. This means that no adversary, regardless of the method applied to the output of a differentially private algorithm, can reduce its privacy guarantees or extract more information about the original dataset. In other words, post-processing cannot make the output less private, providing strong protection against attempts to compromise privacy. While previous work has leveraged this property to improve the utility of the DP-SGD algorithm (Zhang et al., 2024; Balle & Wang, 2018), none have utilized results from random matrix theory for the post-processing function. To our knowledge, this is the first work to apply such results in the context of DP-SGD.

2.2 SINGULAR VALUE DISTRIBUTION OF GRADIENTS

The gradients of linear layers of neural networks in training, when viewed as a linear operator, exhibit a low rank structure (Li et al., 2022), (Zhao et al., 2024). Viewing the singular values of the gradient operator, this translates to a rapid decay in the singular values of the gradient matrix. This is a well known phenomenon in the literature, and has been observed in many different settings, e.g. (Li et al., 2022), (Zhao et al., 2024). While this has been used to explain why differential privacy works so well in deep models with large parameter counts contrary to theoretical expectations (Li et al., 2022), it has not been used to improve the sample efficiency of differentially private training. In this work, we use this property to improve the sample efficiency of differentially private training by using low rank matrix estimation techniques to denoise the gradients before passing them to the optimizer.

2.3 LOW RANK MATRIX ESTIMATIONS

Low rank matrix reconstruction is a rich sub-field of signal processing (Donoho et al., 2018; Gavish & Donoho, 2014; Shabalin & Nobel, 2013). Assuming the rank of the signal matrix $\boldsymbol{X} \in \mathbb{R}^{m \times n}$ is k, we can use the SVD decomposition to write it as

$$\boldsymbol{X} = \sum_{i=1}^{k} \lambda_i \boldsymbol{u}_i \boldsymbol{v}_i^T$$

where λ_is are non-increasing singular values, and $\boldsymbol{u}_i \in \mathbb{R}^m$, $\boldsymbol{v}_i \in \mathbb{R}^n$ are orthonormal vectors.

Then, a noise matrix with entries drawn from $\mathcal{N}(0, \sigma^2)$ is added to get the noisy matrix $\tilde{\boldsymbol{X}}$:

$$\tilde{\boldsymbol{X}} = \boldsymbol{X} + \boldsymbol{\Delta}, \quad \boldsymbol{\Delta}_{ij} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2)$$

The goal is to estimate the original matrix $\boldsymbol{X}$ from the noisy observation $\tilde{\boldsymbol{X}}$. We write the SVD decomposition of $\tilde{\boldsymbol{X}}$ in the notation

$$\tilde{\boldsymbol{X}} = \sum_{i=1}^{\min(m,n)} \tilde{\lambda}_i \tilde{\boldsymbol{u}}_i \tilde{\boldsymbol{v}}_i^T$$

Note that the noisy version may (and usulaly does) have more than k non-zero components.

2.3.1 Effect of noise on singular values and singular vectors: a phase transition

With the mentioned notation we have

$$\tilde{\lambda}_i \approx \begin{cases} F_{\sigma,n,m}(\lambda_i) = \sqrt{\left(\lambda_i + \frac{\sigma^2 n}{\lambda_i}\right)\left(\lambda_i + \frac{\sigma^2 m}{\lambda_i}\right)} & \text{if } \lambda_i > \sigma\sqrt[4]{mn} \\ \sigma(\sqrt{m}+\sqrt{n}) & \text{if } \lambda_i \leq \sigma\sqrt[4]{mn} \end{cases} \tag{3}$$

This is an increase in the value of the singular value, which is usual in random matrix theory. It is important to note that these results are typically stated in the asymptotic regime, where the matrix dimensions grow to infinity and the noise variance may scale with the dimensions, often under specific assumptions on the ratio m/n. In practical, finite-dimensional settings, these approximations may incur some error. The precise rate of this error in finite dimensions is not addressed here and could be an interesting direction for further study. The derivation of these results from their asymptotic forms is postponed to the appendix A.

Also, assuming all of the eignvalues of $\boldsymbol{X}$ are distinct, and if $\lambda_i > \sigma\sqrt[4]{mn}$, following Lemma 3 of Gavish & Donoho (2014) or proposition 9 of Shabalin & Nobel (2013), we can write

$$|\langle \boldsymbol{u}_i, \tilde{\boldsymbol{u}}_j \rangle|^2 \approx \begin{cases} \dfrac{\lambda_i^4 - mn\sigma^4}{\lambda_i^4 + m\lambda_i^2\sigma^2} & i = j \\ 0 & i \neq j \end{cases}, \tag{4}$$

and

$$|\langle \boldsymbol{v}_i, \tilde{\boldsymbol{v}}_j \rangle|^2 \approx \begin{cases} \dfrac{\lambda_i^4 - mn\sigma^4}{\lambda_i^4 + n\lambda_i^2\sigma^2} & i = j \\ 0 & i \neq j \end{cases}. \tag{5}$$

However if $\lambda_i \leq \sigma\sqrt[4]{mn}$, then

$$|\langle \boldsymbol{u}_i, \tilde{\boldsymbol{u}}_j \rangle|^2 \approx |\langle \boldsymbol{v}_i, \tilde{\boldsymbol{v}}_j \rangle|^2 \approx 0 \tag{6}$$

2.3.2 Matrix Denoising

Matrix denoising methods aim to recover the underlying signal matrix $\boldsymbol{X}$ from its noisy observation $\tilde{\boldsymbol{X}}$ by leveraging the low-rank structure of the signal. Many of these methods shrink the singular values of the noisy matrix. One such approach is the so-called optimal method discussed in Shabalin & Nobel (2013); Donoho et al. (2018), which outputs a low-rank matrix.

Optimal Denoising The mentioned optimal estimator for the signal matrix can be written as

$$\hat{\boldsymbol{X}}_{\text{optimal}} = \sum_{i=1}^{r} \eta_i \tilde{\boldsymbol{u}}_i \tilde{\boldsymbol{v}}_i^T \tag{7}$$

where, following Shabalin & Nobel (2013), the optimal coefficients are

$$\eta_i = \hat{\lambda}_i \cdot \sqrt{\frac{\hat{\lambda}_i^4 - mn\sigma^4}{\hat{\lambda}_i^4 + m\hat{\lambda}_i^2\sigma^2}} \cdot \sqrt{\frac{\hat{\lambda}_i^4 - mn\sigma^4}{\hat{\lambda}_i^4 + n\hat{\lambda}_i^2\sigma^2}}, \quad \text{where} \quad \hat{\lambda}_i = F^{-1}_{\sigma,n,m}(\tilde{\lambda}_i) \tag{8}$$

for all i such that $\tilde{\lambda}_i > \sigma(\sqrt{m} + \sqrt{n})$, and zero otherwise. It has been shown to achieve the best possible mean squared error (MSE) under certain conditions, particularly when the noise is Gaussian and the signal is low-rank.

3 METHODOLOGY

In this section, we introduce our post-processing method, which leverages equations 7 and 8 to denoise the gradients produced by DP-SGD before they are passed to the optimizer. Note that some of the performance loss in DP-SGD, compared to non-private training, is due to gradient clipping (Bu et al., 2023). However, this work does not address the performance drop from clipping; we focus exclusively on mitigating the loss caused by the added noise.

3.1 FRAMEWORK

We apply the denoising method by aiming to increase the alignment between the denoised gradient and the clipped gradient. Specifically, our objective is to construct a denoising function that, given the noisy gradient as input, produces an output that is more closely aligned with the clipped gradient. Using the notation from Section 2.1.2, we seek a denoising function $\text{Denoise}(\cdot)$ such that

$$\cos(\text{Denoise}(\tilde{\boldsymbol{g}}), \bar{\boldsymbol{g}}) > \cos(\tilde{\boldsymbol{g}}, \bar{\boldsymbol{g}})$$

where $\cos(\boldsymbol{a}, \boldsymbol{b}) = \frac{\boldsymbol{a}^T\boldsymbol{b}}{\|\boldsymbol{a}\|_2\|\boldsymbol{b}\|_2}$ is the cosine similarity between two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$.

For tracking this value, we define the Improvement at step t as

$$\text{Improvement}(t) = \cos(\text{Denoise}(\tilde{\boldsymbol{g}}_t), \bar{\boldsymbol{g}}_t) - \cos(\tilde{\boldsymbol{g}}_t, \bar{\boldsymbol{g}}_t)$$

If we can come up with such a denoising function, we hope to improve the sample efficiency of DP-SGD by making the noisy gradients more closely resemble the true (clipped) gradients. Having such a denoising function, we can change the DP-SGD algorithm as follows:

Algorithm 2 DP-SGD with Denoising

Require: Dataset D, loss function $f(\theta, x)$, model parameters θ, sampling rate ρ, clipping norm C, noise multiplier σ, optimizer $\mathcal{O}$, number of steps T, Denoising function $\text{Denoise}(\cdot)$

for $t \leftarrow 1$ to T **do**
  Sample a batch $\mathcal{B}$ from D using Poisson sampling with rate ρ
  for each $i \in \mathcal{B}$ **do**
    Compute per-example gradient $\boldsymbol{g}_i \leftarrow \nabla_\theta f(\theta, x_i)$
    Clip gradient: $\text{clip}(\boldsymbol{g}_i, C) \leftarrow \boldsymbol{g}_i / \max(1, \|\boldsymbol{g}_i\|_2 / C)$
  end for
  Aggregate clipped gradients: $\bar{\boldsymbol{g}} \leftarrow \sum_{i \in \mathcal{B}} \text{clip}(\boldsymbol{g}_i, C)$
  Draw noise vector $\boldsymbol{w}$ with i.i.d. entries from $\mathcal{N}(0, \sigma^2 C^2)$
  Compute noisy average: $\tilde{\boldsymbol{g}} \leftarrow (\bar{\boldsymbol{g}} + \boldsymbol{w})/|\mathcal{B}|$
  Denoise the gradient: $\hat{\boldsymbol{g}} \leftarrow \text{Denoise}(\tilde{\boldsymbol{g}})$
  Update optimizer state: $\mathcal{O} \leftarrow \text{UpdateState}(\mathcal{O}, \hat{\boldsymbol{g}})$
  Update parameters: $\theta \leftarrow \text{UpdateParameters}(\theta, \mathcal{O})$
end for

We expect that if the improvement at each step t is consistently non negative, $\text{Improvement}(t) \geq 0$, then the denoising function is effectively aligning the noisy gradients with the true (clipped) gradients, leading to faster convergence of the DP-SGD algorithm. The following sections will detail the implementation of the denoising function which are mainly based on the results reviewed in Section 2.3.2.

3.2 DENOISING FUNCTION

The denoising function we propose is basically application of the denoising functions in section 2.3.2 to the linear components of the noisy gradient $\tilde{\boldsymbol{g}}$. Supposing $\boldsymbol{W}$ is a layer of our neural network θ, the restriction of the (clipped) gradient to $\boldsymbol{W}$ is a matrix $\sum_{x\in\mathcal{B}} \text{clip}(\nabla_\theta f(\theta, x), C)\,|_{\boldsymbol{W}} = \bar{\boldsymbol{g}}\,|_{\boldsymbol{W}}$. If we consider all the different layers of the neural network, the parameters of the neural network can be partitioned as

$$\theta = \boldsymbol{W}_1 \times \boldsymbol{W}_2 \times \ldots \times \boldsymbol{W}_L$$

where L is the number of layers in the network. Then, we can write

$$\bar{\boldsymbol{g}} = (\bar{\boldsymbol{g}}\,|_{W_1}, \bar{\boldsymbol{g}}\,|_{W_2}, \ldots, \bar{\boldsymbol{g}}\,|_{W_L})$$
$$\tilde{\boldsymbol{g}} = (\tilde{\boldsymbol{g}}\,|_{W_1}, \tilde{\boldsymbol{g}}\,|_{W_2}, \ldots, \tilde{\boldsymbol{g}}\,|_{W_L})$$

With this notation, we can define the denoising function as seperate application of the denoising functions to each layer's gradient:

$$\text{Denoise}(\tilde{\boldsymbol{g}}) = (\text{Denoise}(\tilde{\boldsymbol{g}}\,|_{W_1}), \text{Denoise}(\tilde{\boldsymbol{g}}\,|_{W_2}), \ldots, \text{Denoise}(\tilde{\boldsymbol{g}}\,|_{W_L}))$$

Where if a layer $\boldsymbol{W}_i$ is not a linear layer, we simply set $\text{Denoise}(\tilde{\boldsymbol{g}}_{W_i}) = \tilde{\boldsymbol{g}}_{W_i}$.

For linear layers, we modify the so called "optimal" denoising method in two ways

- Only applying gradient if the singular values of the noisy layer gradient are larger than a preset multiple of $\sigma(\sqrt{n} + \sqrt{m})$, the largest singular value of the bulk.
- When the singular value is larger than the required threshold, we apply the optimal denoising function, then rescale it so that its length is equal to the noisy version $\tilde{\boldsymbol{g}} \mapsto \frac{||\tilde{\boldsymbol{g}}||}{||\hat{\boldsymbol{g}}_{\text{optimal}}||}\hat{\boldsymbol{g}}_{\text{optimal}}$.

So for linear layers and the hyperparameter κ we have

$$\text{Denoise}(\tilde{\boldsymbol{g}}\,|_{\boldsymbol{W}}) = \begin{cases} \tilde{\boldsymbol{g}}\,|_{\boldsymbol{W}}, & \text{if } \lambda_1(\tilde{\boldsymbol{g}}\,|_{\boldsymbol{W}}) < \kappa\,\sigma(\sqrt{n} + \sqrt{m}) \\ \frac{||\tilde{\boldsymbol{g}}||}{||\hat{\boldsymbol{g}}_{\text{optimal}}||}\hat{\boldsymbol{g}}_{\text{optimal}}, & \text{otherwise} \end{cases}$$

3.2.1 WHY THRESHOLD IS NEEDED?

It is important to recognize that the results in Section 2.3.2 are derived in asymptotic settings. For instance, the theory predicts that if all singular values of the signal matrix are less than $\sigma\sqrt[4]{nm}$, or equivalently, if all singular values of the noisy matrix are less than $\sigma(\sqrt{n} + \sqrt{m})$, then the inner products between the left (or right) singular vectors of the signal and noisy matrices should be zero. In that case, the optimal denosing algorithm returns the zero matrix as the optimum result and states that it is the best one can get. However, in practice and for finite-dimensional matrices, this is not true, and the noisy gradient, even if its singular value are small, usually still has some positive cosine similarity with the original gradient. As a result, the denoising algorithm does not always improve the alignment between the noisy and clipped gradients.

Fortunately, we identified a simple heuristic criterion to guide when the denoising algorithm should be applied. Specifically, we require that the largest singular value of the noisy gradient matrix exceeds the threshold $\kappa\,\sigma(\sqrt{n} + \sqrt{m})$ before applying the denoising algorithm. Our observations show that when this condition is met, denoising tends to improve the alignment; otherwise, it may decrease it. For choosing the value of κ, we tuned it on the SST dataset while training the roberta-base model by choosing the best value from the set $\{1.01, 1.02, 1.05, 1.1\}$, and used the same value for all the other model/datset pairs.

3.2.2 WHY NORM CORRECTION IS NEEDED?

Norm correction is needed because of the possible big reduction in the norm of the denoised gradient compared to the noisy gradient. This may in turn cause the global improvement at time t to become negative. Why this might happen, and how rescaling helps prevent it, will be explained in the appendix B.3, as well as results of an abalation study on the effect of norm correction.

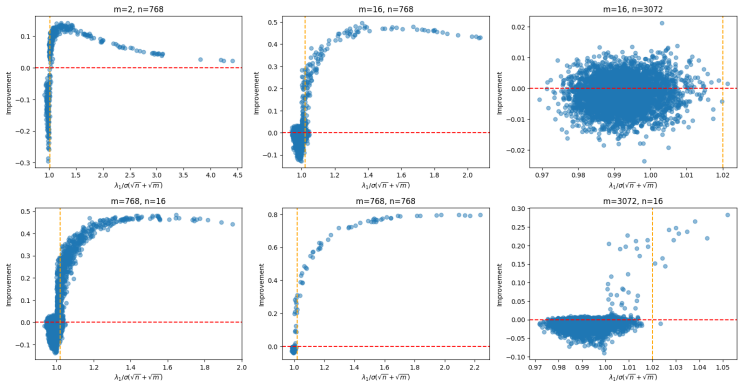

Figure 3: Scatter plot of layer improvement vs $\frac{\lambda_1}{\sigma(\sqrt{n}+\sqrt{m})}$ for different layer dimensionality. The vertical yellow line shows the threshold κ we used in our experiments. We want the yellow line in a position to have lots of points on top right side, and few points on the bottom right side (and preferably few on top left side).

4 Experiments

In this section, we present the evaluation method and the experiment results we had. To evaluate our main goal of improving the sample efficiency of DP-SGD, we compared the performance of DP-SGD with and without our denoising method across different datasets from the GLUE benchmark (Wang et al., 2019) and two sizes of the roberta model (Liu et al., 2019).

Because our goal is to find a fast converging method, with possible trade-off in the final performance, we count the number of training steps each method needs to reach some certain (validation) accuracy thresholds. We set these thresholds to be 95% and 90% of the SOTA results for the private training of the same models on the same datasets. The SOTA results are taken from Yu et al. (2021).

For epsilon, we also follow the same setup as Yu et al. (2021), which is 6.7 for all datasets, and compute the required noise multiplier using the privacy accountant of Gopi et al. (2021) in each case so that the total privacy loss at 400 steps is 6.7.

We keep every other hyper-parameter the same as Yu et al. (2021), including batch size, learning rate, weight decay, and clipping norm. Looking at the tables 1 and 2, we can see that our method consistently improves the sample efficiency of DP-SGD across all datasets and model sizes. Improvements range from 20% to 100% in the number of steps required to reach 90% and 95% of the SOTA performance. Also, we achieved higher performance in five out of eight cases for the final accuracy at 400 steps.

To explain why this method converges faster, we can check the improvement in cosine similarity between the denoised and noisy gradients with respect to the clipped gradients. As shown in Figure 4, the denoising method consistently improves the alignment between the noisy and clipped gradients throughout the training process, which aligns with our explanation of why the speedup happens.

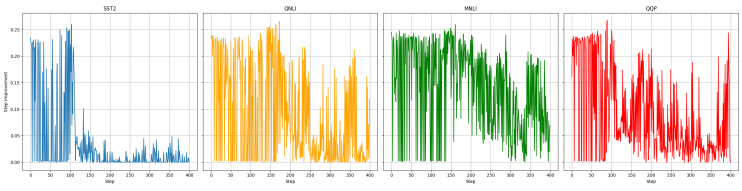

Figure 4: Improvement in cosine similarity between denoised and noisy gradients with respect to clipped gradients over training steps for different datasets. The positive values indicate that the denoising method consistently enhances the alignment between the noisy and clipped gradients throughout the training process.

Task	Method	Final Acc. (at 400 steps)	SOTA (at 20 epochs)	Steps 90%	Steps 95%	Speedup 90%	Speedup 95%
SST	Ours	92.4	92.5	150	150	67%	67%
	Baseline	**92.5**		250	250		
QNLI	Ours	**84.6**	87.5	200	300	100%	-
	Baseline	80.0		400	–		
MNLI	Ours	**80.0**	83.5	250	400	40%	-
	Baseline	77.6		350	–		
QQP	Ours	**83.1**	85.7	150	250	67%	40%
	Baseline	81.9		250	350		

Table 1: Comparison of Ours and Baseline on GLUE tasks when training Roberta Base. Final accuracy, SOTA reference, number of steps needed to reach 90% and 95% of SOTA, and speedups (only for Ours) are reported.

Task	Method	Final Acc. (at 400 steps)	SOTA (at 20 epochs)	Steps 90%	Steps 95%	Speedup 90%	Speedup 95%
SST	Ours	93.8	95.3	150	150	33%	67%
	Baseline	**93.9**		200	250		
QNLI	Ours	88.5	90.8	150	250	33%	20%
	Baseline	**89.2**		200	300		
MNLI	Ours	**85.6**	87.8	200	250	25%	20%
	Baseline	85.3		250	300		
QQP	Ours	**84.7**	87.4	150	250	33%	20%
	Baseline	84.1		200	300		

Table 2: Comparison of Ours and Baseline on GLUE tasks with RoBERTa Large. Final accuracy, SOTA reference, steps to reach 90% and 95% of SOTA, and speedups (only for Ours) are reported.

5 Limitations and Future Work

One major limitation of our method is that it is does not always produces the best performance as well as the fastest convergence. In some of the experiments, the baseline method achieves slightly better final accuracy than our method. This is specially puzzling because of the dominantly positive improvement in cosine similarity between the denoised and noisy gradients with respect to the clipped gradients. This calls for further investigation to understand why this happens, and how to fix it.

6 Reproducibility Statement

All the necessary code and hyperparameters for reproducing the results in this paper has been made available in the supplementary material.

References

Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 308–318, 2016.

Jinho Baik and Jack W Silverstein. Eigenvalues of large sample covariance matrices of spiked population models. *Journal of multivariate analysis*, 97(6):1382–1408, 2006.

Jinho Baik, Gérard Ben Arous, and Sandrine Péché. Phase transition of the largest eigenvalue for nonnull complex sample covariance matrices. 2005.

Borja Balle and Yu-Xiang Wang. Improving the Gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 394–403. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/balle18a.html.

Florent Benaych-Georges and Raj Rao Nadakuditi. The singular values and vectors of low rank perturbations of large rectangular random matrices. *Journal of Multivariate Analysis*, 111:120–135, 2012.

Zhiqi Bu, Hua Wang, Zongyu Dai, and Qi Long. On the convergence and calibration of deep learning with differential privacy. *Transactions on machine learning research*, 2023:https–openreview, 2023.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. Extracting training data from large language models. In *30th USENIX security symposium (USENIX Security 21)*, pp. 2633–2650, 2021.

Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. Membership inference attacks from first principles. In *2022 IEEE symposium on security and privacy (SP)*, pp. 1897–1914. IEEE, 2022.

David L Donoho, Matan Gavish, and Iain M Johnstone. Optimal shrinkage of eigenvalues in the spiked covariance model. *Annals of statistics*, 46(4):1742, 2018.

Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and trends® in theoretical computer science*, 9(3–4):211–407, 2014.

Matan Gavish and David L Donoho. The optimal hard threshold for singular values is $4/\sqrt{3}$. *IEEE Transactions on Information Theory*, 60(8):5040–5053, 2014.

Sivakanth Gopi, Yin Tat Lee, and Lukas Wutschitz. Numerical composition of differential privacy. *Advances in Neural Information Processing Systems*, 34:11631–11642, 2021.

Xuechen Li, Florian Tramer, Percy Liang, and Tatsunori Hashimoto. Large language models can be strong differentially private learners. *arXiv preprint arXiv:2110.05679*, 2021.

Xuechen Li, Daogao Liu, Tatsunori B Hashimoto, Huseyin A Inan, Janardhan Kulkarni, Yin-Tat Lee, and Abhradeep Guha Thakurta. When does differentially private learning not suffer in high dimensions? *Advances in Neural Information Processing Systems*, 35:28616–28630, 2022.

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

Vladimir A Marčenko and Leonid Andreevich Pastur. Distribution of eigenvalues for some sets of random matrices. *Mathematics of the USSR-Sbornik*, 1(4):457, 1967.

Andrey A Shabalin and Andrew B Nobel. Reconstruction of a low-rank matrix in the presence of gaussian noise. *Journal of Multivariate Analysis*, 118:67–76, 2013.

Terence Tao. *Topics in random matrix theory*, volume 132. American Mathematical Soc., 2012.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*, 2019.

Da Yu, Saurabh Naik, Arturs Backurs, Sivakanth Gopi, Huseyin A Inan, Gautam Kamath, Janardhan Kulkarni, Yin Tat Lee, Andre Manoel, Lukas Wutschitz, et al. Differentially private fine-tuning of language models. *arXiv preprint arXiv:2110.06500*, 2021.

Xinwei Zhang, Zhiqi Bu, Mingyi Hong, and Meisam Razaviyayn. Doppler: Differentially private optimizers with low-pass filter for privacy noise reduction. *Advances in neural information processing systems*, 37:41826–41851, 2024.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. Galore: Memory-efficient llm training by gradient low-rank projection. *arXiv preprint arXiv:2403.03507*, 2024.

A Finite dimensional derivation of random matrix theory results

In the usual random matrix theory literature, the results in Shabalin & Nobel (2013); Donoho et al. (2018); Gavish & Donoho (2014) are stated in the asymptotic regime, where the matrix dimensions grow to infinity and the noise variance may scale with the dimensions. In this section we want to state those results in their original form, and explain the derivation of equations 3, 4, and 5 from their asymptotic forms.

The setup in Shabalin & Nobel (2013); Donoho et al. (2018) is as follows. We have a sequence of matrices $\boldsymbol{X}_n \in \mathbb{R}^{m_n \times n}$ with $m_n/n \to \beta$ as $n \to \infty$. The rank of the signal matrix is fixed, i.e. $\mathrm{rank}(\boldsymbol{X}_n) = r$ for all n. The singular values of the signal matrix are fixed, i.e. the non-zero singular values of $\boldsymbol{X}_n$ are $\lambda_1 > \lambda_2 > \ldots > \lambda_r > 0$ for all n. Then, we add a noise matrix with i.i.d. entries from $\mathcal{N}(0, 1/n)$ to get the noisy matrix. In these settings, the results in Shabalin & Nobel (2013); Gavish & Donoho (2014) state that

$$\lim_{n\to\infty} y_{n,i} \overset{a.s.}{=} \begin{cases} \sqrt{\left(\lambda_i + \frac{1}{\lambda_i}\right)\left(\lambda_i + \frac{\beta}{\lambda_i}\right)} & \lambda_i > \beta^{1/4} \\ 1 + \sqrt{\beta} & \lambda_i \leq \beta^{1/4} \end{cases} \tag{9}$$

where $y_{n,i}$ is the i-th singular value of the noisy matrix. If we want to change this into the finite dimensional form, we can start form a noise matrix with i.i.d. entries from $\mathcal{N}(0, \sigma^2)$ instead of $\mathcal{N}(0, 1/n)$. Then, if we work with the matrix $\frac{Y}{\sigma\sqrt{n}}$, then, the new noise matrix will have the desired distribution. Using the equation 9 for the matrix $\frac{Y}{\sigma\sqrt{n}}$, and substituting $\beta = m/n$, we get to the equation 3. Similar arguments can be used to derive equations 4 and 5 from their asymptotic forms in Shabalin & Nobel (2013).

B Why norm correction is needed?

We want to show in this section that why improving cosine similarity of one component of a noisy vector to the signal may not necassiryly improve the overall cosine similarity, and why norm correction helps prevent this issue.

B.1 Setup and Definitions

Let vectors be partitioned into two components:

$$a = (a_1, a_2), \quad b = (b_1, b_2),$$

with $a_1, b_1 \in \mathbb{R}^{d_1}$ and $a_2, b_2 \in \mathbb{R}^{d_2}$.

We define the following concepts for vectors partitioned into two components as above.

Overall cosine similarity: The cosine similarity between a and b is

$$\cos(a, b) = \frac{a_1 \cdot b_1 + a_2 \cdot b_2}{\|a\| \, \|b\|}.$$

Block cosine similarity: The cosine similarity between the second blocks is

$$\cos(a_2, b_2) = \frac{a_2 \cdot b_2}{\|a_2\| \, \|b_2\|}.$$

Improvement of block similarity: Given a modified block a_2', we say that a_2' improves the similarity of block 2 with respect to b_2 if

$$\cos(a_2', b_2) > \cos(a_2, b_2).$$

We define the modified full vector as $a' = (a_1, a_2')$.

B.2 Improving a block is not sufficient globally

It is possible that $\cos(a_2', b_2) > \cos(a_2, b_2)$ but

$$\cos(a', b) < \cos(a, b).$$

Proof. We can write

$$\cos(a, b) = \frac{a_1 \cdot b_1 + \|a_2\|\|b_2\| \cos\theta}{\sqrt{\|a_1\|^2 + \|a_2\|^2} \, \|b\|},$$
$$\cos(a', b) = \frac{a_1 \cdot b_1 + \|a_2'\|\|b_2\| \cos\theta'}{\sqrt{\|a_1\|^2 + \|a_2'\|^2} \, \|b\|},$$

where θ, θ' are the angles between a_2, b_2 and a_2', b_2.

Even if $\cos\theta' > \cos\theta$, the numerator can decrease when $\|a_2'\| < \|a_2\|$. Since the denominator also changes with $\|a_2'\|$, it is possible that $\cos(a', b) < \cos(a, b)$. Explicit counterexamples confirm this. □

B.3 Equal block norm guarantees improvement

If $\|a_2'\| = \|a_2\|$ and $\cos(a_2', b_2) > \cos(a_2, b_2)$, then

$$\cos(a', b) > \cos(a, b).$$

Proof. Let $r = \|a_2\| = \|a_2'\|$ and $s = \|b_2\|$. Then

$$a_2 \cdot b_2 = rs\cos\theta, \qquad a_2' \cdot b_2 = rs\cos\theta', \qquad \cos\theta' > \cos\theta.$$

Numerators:

$$N = a_1 \cdot b_1 + rs\cos\theta, \qquad N' = a_1 \cdot b_1 + rs\cos\theta',$$

so $N' > N$.

Denominators:

$$\|a\| = \sqrt{\|a_1\|^2 + r^2} = \|a'\|, \qquad \|b\| \text{ fixed.}$$

Therefore

$$\cos(a', b) = \frac{N'}{\|a\|\|b\|} > \frac{N}{\|a\|\|b\|} = \cos(a, b).$$

□

C Use of LLMs

We have utilized large language models (LLMs) to assist in editing and refining the manuscript. LLMs were used to improve the clarity, coherence, and overall quality of the writing, ensuring that the content is presented in a clear and accessible manner.