DO NOT LET PRIVACY OVERBILL UTILITY: GRADIENT EMBEDDING PERTURBATION FOR PRIVATE LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

The privacy leakage of the model about the training data can be bounded in the differential privacy mechanism. However, for meaningful privacy parameters, a differentially private model degrades the utility drastically when the model comprises a large number of trainable parameters. In this paper, we propose an algorithm *Gradient Embedding Perturbation (GEP)* towards training differentially private deep models with decent accuracy. Specifically, in each gradient descent step, GEP first projects individual private gradient into a non-sensitive anchor subspace, producing a low-dimensional gradient embedding and a small-norm residual gradient. Then, GEP perturbs the low-dimensional embedding and the residual gradient separately according to the privacy budget. Such a decomposition permits a small perturbation variance, which greatly helps to break the dimensional barrier of private learning. With GEP, we achieve decent accuracy with reasonable computational cost and modest privacy guarantee for deep models. Especially, with privacy bound $\epsilon = 8$, we achieve 74.9% test accuracy on CIFAR10 and 95.1% test accuracy on SVHN, significantly improving over existing results.

1 INTRODUCTION

Recent works have shown that the trained model may leak/memorize the information of its training set (Fredrikson et al., 2015; Wu et al., 2016; Shokri et al., 2017; Hitaj et al., 2017), which raises privacy issue when the model are trained with sensitive data. *Differential privacy* (DP) mechanism provides a way to quantitatively measure and upper bound such information leakage. It theoretically ensures that the influence of any individual sample is negligible with the DP parameter ϵ or (ϵ, δ). Moreover, it has been observed that differentially private models can also resist model inversion attack (Carlini et al., 2019), membership inference attack (Rahman et al., 2018; Bernau et al., 2019), gradient matching attack (Zhu et al., 2019), and data poisoning attack (Ma et al., 2019).

One popular way to achieve differentially private machine learning is to perturb the training process with noise (Song et al., 2013; Bassily et al., 2014; Shokri & Shmatikov, 2015; Wu et al., 2017; Fukuchi et al., 2017; Iyengar et al., 2019; Phan et al., 2020). Specifically, *gradient perturbation* perturbs the gradient at each iteration of (stochastic) gradient descent algorithm and guarantees the privacy of the final model via *composition property* of DP. It is worthy to note that gradient perturbation does not assume (strongly) convex objective and hence is applicable to various settings (Abadi et al., 2016; Wang et al., 2017; Lee & Kifer, 2018; Jayaraman et al., 2018; Wang & Gu, 2019). Specifically, for given gradient sensitivity S, a general form of gradient perturbation is to add an isotropic Gaussian noise $\boldsymbol{z}$ to the gradient $\boldsymbol{g} \in \mathbb{R}^p$ independently for each step,

$$\tilde{\boldsymbol{g}} = \boldsymbol{g} + \boldsymbol{z}, \quad \text{where} \quad \boldsymbol{z} \sim \mathcal{N}(0, \sigma^2 S^2 \boldsymbol{I}_{p\times p}). \tag{1}$$

One can set proper variance σ^2 to make each update differentially private with parameter (ϵ, δ). It is easy to see that the intensity of the added noise $\mathbb{E}[\|\boldsymbol{z}\|^2]$ scales linearly with the model dimension p. This indicates that as the model becomes larger, the useful signal, i.e., gradient, would be submerged in the added noise (see Figure 1). This dimensional barrier restricts the utility of deep learning models trained with gradient perturbation.

The dimensional barrier is attributed to the fact that the added noise is isotropic while the gradients live on a very low dimensional manifold, which has been observed in (Gur-Ari et al., 2018; Vogels

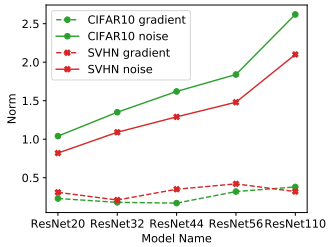

Figure 1: Noise norm vs gradient norm of ResNet20 at initialization. The noise variance is chosen such that SGD satisfies $(5, 10^{-5})$-DP after 90 epochs in Abadi et al. (2016).

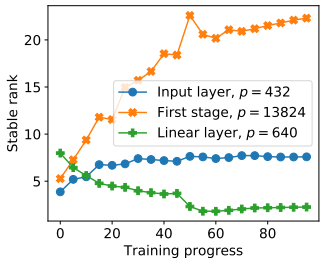

Figure 2: Stable rank $\|\cdot\|_F/\|\cdot\|$ of batch gradient matrix of given groups (with p parameters). The setting is ResNet20 on CIFAR-10. The stable rank is small throughout training.

et al., 2019; Gooneratne et al., 2020; Li et al., 2020) and is also verified in Figure 2 for the gradients of a 20-layer ResNet (He et al., 2016). Hence to limit the noise energy, it is natural to think

"Can we reduce the dimension of gradients first and then add the isotropic noise onto a low-dimensional gradient embedding?"

The answer is affirmative. We propose a new algorithm *Gradient Embedding Perturbation (GEP)*, illustrated in Figure 3. Specifically, we first compute *anchor gradients* on some non-sensitive auxiliary data, and identify an *anchor subspace* that is spanned by several top principal components of the anchor gradient matrix. Then we project the private gradients into the anchor subspace and obtain low-dimensional *gradient embeddings* and small-norm *residual gradients*. Finally, we perturb the gradient embedding and residual gradient separately according to the sensitivities and privacy budget.

We intuitively argue why GEP could reduce the perturbation variance and achieve good utility for large models. First, because the gradient embedding has a very low dimension, the added isotropic noise on embedding has small energy that scales linearly only with the subspace dimension. Second, if the anchor subspace can cover most of the gradient information, the residual gradient, though high dimensional, should have small magnitude, which permits smaller added noise to guarantee the same level privacy because of the reduced sensitivity. Overall, we can use a much lower perturbation compared with the original gradient perturbation to guarantee the same level of privacy.

We emphasize several properties of GEP. First, the non-sensitive auxiliary data assumption is weak. In fact, GEP only requires a small number of non-sensitive unlabeled data following a similar feature distribution as the private data, which often exist even for learning on sensitive data. In our experiments, we use a few unlabeled samples from ImageNet to serve as auxiliary data for MNIST, SVHN, and CIFAR-10. This assumption is much weaker than the public data assumption in previous works (Papernot et al., 2017; 2018; Alon et al., 2019; Wang & Zhou, 2020), where the public data should follow exactly the same distribution as the private data. Second, GEP produces an unbiased estimator of the target gradient because of releasing both the perturbed gradient embedding and the perturbed residual gradient, which turns out to be critical for good utility. Third, we use *power method* to estimate the principal components of anchor gradients, achievable with a few matrix multiplications. The fact that GEP is not sensitive to the choices of subspace dimension further allows a very efficient implementation.

Compared with existing works of differentially private machine learning, our contribution can be summarized as follows: (1) we propose a novel algorithm GEP that achieves good utility for large models with modest differential privacy guarantee; (2) we show that GEP returns an unbiased estimator of target private gradient with much lower perturbation variance than original gradient perturbation; (3) we demonstrate that GEP achieves state-of-the-art utility in differentially private learning with three benchmark datasets. Specifically, for $\epsilon = 8$, GEP achieves 74.9% test accuracy on CIFAR-10 with a ResNet20 model. To the best of our knowledge, GEP is the first algorithm that can achieve such utility with training deep models from scratch for a "single-digit" privacy budget[1].

[1] Abadi et al. (2016) achieve 73% accuracy on CIFAR-10 but they need to pre-train the model on CIFAR-100.

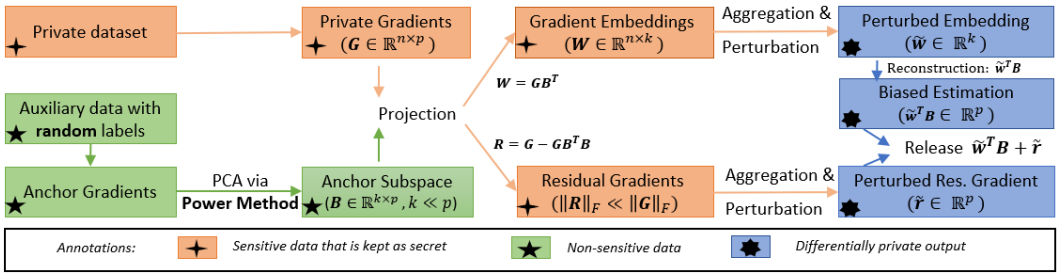

Figure 3: Overview of the proposed GEP approach. 1) We estimate an anchor subspace on some non-sensitive data; 2) We project the private gradients into the anchor subspace, producing low-dimensional embeddings and residual gradients; 3) We perturb the gradient embedding and residual gradient separately to guarantee differential privacy. The auxiliary data are only required to share similar features as the private data. In our experiments, we use 2000 images from ImageNet as auxiliary data for MNIST, SVHN, and CIFAR-10 datasets.

1.1 Related work

Existing works studying differentially private machine learning in high-dimensional setting can be roughly categorized into two sets. One is treating the optimization of the machine learning objective as a whole mechanism and adding noise into this process. The other one is based on the knowledge transfer of machine learning models, which trains a differentially private publishable student model with private signals from teacher models. We review them one by one.

Differentially private convex optimization in high-dimensional setting has been studied extensively over the years (Kifer et al., 2012; Thakurta & Smith, 2013; Talwar et al., 2015; Wang & Xu, 2019; Wang & Gu, 2019). Although these methods demonstrate good utility on some convex settings, their analyses can not be directly applied to non-convex setting. Right before the submission, we note two independent and concurrent works (Zhou et al., 2020; Kairouz et al., 2020) that also leverage the gradient redundancy to reduce the added noise. Specifically, Kairouz et al. (2020) track historical gradients to do dimension reduction for private AdaGrad. Zhou et al. (2020) requires gradients on some public data and then project the noisy gradients into a public subspace at each update. One core difference between these two works and GEP is that we introduce residual gradient perturbation and GEP produces an unbiased estimator of the private gradients, which is essential for achieving the superior utility. Moreover, we weaken the auxiliary data assumption and introduce several designs that significantly boost the efficiency and applicability of GEP.

One recent progress towards training arbitrary models with differential privacy is *Private Aggregation of Teacher Ensembles (PATE)* (Papernot et al., 2017; 2018; Jordon et al., 2019). PATE first trains independent teacher models on disjoint shards of private data. Then it trains a student model with privacy guarantee by distilling noisy predictions of teacher models on some public samples. In comparison, GEP only requires some non-sensitive data that have similar natural features as the private data while PATE requires the public data follow exactly the same distribution as the private data and in practice it uses a portion of the test data to serve as public data. Moreover, GEP demonstrates better performance than PATE especially for complex datasets, e.g., CIFAR-10, because GEP can train the model with the whole private data rather than a small shard of data.

2 Preliminaries

We introduce some notations and definitions. We use bold lowercase letters, e.g., $\boldsymbol{v}$, and bold capital letters, e.g., $\boldsymbol{M}$, to denote vectors and matrices, respectively. The L^2 norm of a vector $\boldsymbol{v}$ is denoted by $\|\boldsymbol{v}\|$. The spectral norm and the Frobenius norm of a matrix $\boldsymbol{M}$ are denoted by $\|\boldsymbol{M}\|$ and $\|\boldsymbol{M}\|_F$, respectively. A sample $d = (\boldsymbol{x}, y)$ consists of feature $\boldsymbol{x}$ and label y. A dataset $\mathbb{D}$ is a collection of individual samples. A dataset $\mathbb{D}'$ is said to be a neighboring dataset of $\mathbb{D}$ if they differ in a single sample, denoted as $\mathbb{D} \sim \mathbb{D}'$. Differential privacy ensures that the outputs of an algorithm on neighboring datasets have approximately indistinguishable distributions.

Definition 1 ((ϵ, δ)-DP (Dwork et al., 2006a;b)). *A randomized mechanism $\mathcal{M}$ guarantees (ϵ, δ)-differential privacy if for any two neighboring input datasets $\mathbb{D} \sim \mathbb{D}^{'}$ and for any subset of outputs S it holds that $Pr[\mathcal{M}(\mathbb{D}) \in S] \leq e^{\epsilon} Pr[\mathcal{M}(\mathbb{D}^{'}) \in S] + \delta$.*

By its definition, (ϵ, δ)-DP controls the maximum influence that any individual sample can produce. One can adjust the privacy parameters to trade off between privacy and utility. Differential privacy is immune to *post-processing* (Dwork et al., 2014), i.e., any function applied on the output of a differentially private algorithm would not increase the privacy loss as long as it does not have new interaction with the private dataset. Differential privacy also allows *composition*, i.e., the composition of a series of differentially private mechanisms is also differentially private but with different parameters. Several variants of (ϵ, δ)-DP have been proposed (Bun & Steinke, 2016; Dong et al., 2019) to address certain weakness of (ϵ, δ)-DP, e.g., they achieve better composition property. In this work, we use *Rényi differential privacy* (Mironov, 2017) to track the privacy loss and then convert it to (ϵ, δ)-DP.

Suppose that there is a private dataset $\mathbb{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ with n samples. We want to train a model f to learn the mapping in $\mathbb{D}$. Specifically, f takes $\boldsymbol{x}$ as input and outputs a label y, and f has parameter $\theta \in \mathbb{R}^p$. The training objective is to minimize an empirical risk $\frac{1}{n}\sum_{i=1}^n \ell(f(\boldsymbol{x}_i), y_i)$, where $\ell(\cdot, \cdot)$ is a loss function. We further assume that there is an auxiliary dataset $\mathbb{D}^{(a)} = \{(\tilde{\boldsymbol{x}}_j, \tilde{y}_j)\}_{j=1}^m$ that $\tilde{\boldsymbol{x}}$ shares similar features as $\boldsymbol{x}$ in $\mathbb{D}$ while $\tilde{y}$ could be random.

3 GRADIENT EMBEDDING PERTURBATION

An overview of GEP is given in Figure 3. GEP has three major ingredients: 1) first, estimate an anchor subspace that contains the principal components of some non-sensitive anchor gradients via power method; 2) then, project private gradients into the anchor subspace and produce low-dimensional embeddings of private gradients and residual gradients; 3) finally, perturb gradient embedding and residual gradient separately to establish differential privacy guarantee. In Section 3.1, we present the GEP algorithm in detail. In Section 3.2, we given an analysis on the residual gradients. In Section 3.3, we give a differentially private learning algorithm that updates the model with the output of GEP.

3.1 THE GEP ALGORITHM AND ITS PRIVACY ANALYSIS

The pseudocode of GEP is presented in Algorithm 1. For convenience, we write a set of gradients and a set of basis vectors as matrices with each row being one gradient/basis vector.

The anchor subspace is constructed as follows. We first compute the gradients of the model on an auxiliary dataset $\mathbb{D}^{(a)}$ with m samples, which is referred to as the anchor gradients $\boldsymbol{G}^{(a)} \in \mathbb{R}^{m \times p}$. We then use the power method to estimate the principal components of $\boldsymbol{G}^{(a)}$ to construct a subspace basis $\boldsymbol{B} \in \mathbb{R}^{k \times p}$, which is referred to as the anchor subspace. All these matrices are publishable because $\mathbb{D}^{(a)}$ is non-sensitive. We expect that the anchor subspace $\boldsymbol{B}$ can cover most energy of private gradients when the auxiliary data are not far from private data and m, k are reasonably large.

Suppose that the private gradients are $\boldsymbol{G} \in \mathbb{R}^{n \times p}$. Then, we project the private gradients into the anchor subspace $\boldsymbol{B}$. The projection produces low-dimensional embeddings $\boldsymbol{W} = \boldsymbol{G}\boldsymbol{B}^T$ and residual gradients $\boldsymbol{R} = \boldsymbol{G} - \boldsymbol{G}\boldsymbol{B}^T\boldsymbol{B}$. The magnitude of residual gradients is usually much smaller than original gradient even when k is small because of the gradient redundancy.

Then, we aggregate the gradient embeddings and the residual gradients, respectively. We perturb the aggregated embedding and the aggregated residual gradient respectively to guarantee certain differential privacy. Finally, we release the perturbed embedding and the perturbed residual gradient and construct an unbiased estimator of the private gradient: $\tilde{\boldsymbol{v}} := (\tilde{\boldsymbol{w}}^T\boldsymbol{B} + \tilde{\boldsymbol{r}})/n$. This construction process does not resulting in additional privacy loss because of DP's post-processing property. The privacy analysis of the whole process of GEP is given in Theorem 3.1.

Theorem 3.1. *Let S_1 and S_2 be the sensitivity of $\boldsymbol{w}$ and $\boldsymbol{r}$, respectively, the output of Algorithm 1 satisfies (ϵ, δ)-DP for any $\delta \in (0, 1)$ and $\epsilon \leq 2\log(1/\delta)$ if we choose $\sigma_1 \geq 2S_1\sqrt{2\log(1/\delta)}/\epsilon$ and $\sigma_2 \geq 2S_2\sqrt{2\log(1/\delta)}/\epsilon$.*

Algorithm 1: Gradient embedding perturbation

Input: anchor gradients $\boldsymbol{G}^{(a)} \in \mathbb{R}^{m\times p}$; number of basis vectors k; private gradients $\boldsymbol{G} \in \mathbb{R}^{n\times p}$; clipping thresholds S_1, S_2; standard deviations σ_1, σ_2; number of power iterations t.

//First stage: Compute an orthonormal basis for the anchor subspace.
Initialize $\boldsymbol{B} \in \mathbb{R}^{k\times p}$ randomly.
for $i = 1$ **to** t **do**
  Compute $\boldsymbol{A} = \boldsymbol{G}^{(a)}\boldsymbol{B}^T$ and $\boldsymbol{B} = \boldsymbol{A}^T\boldsymbol{G}^{(a)}$.
  Orthogonalize $\boldsymbol{B}$ and normalize row vectors.
end for
Delete $\boldsymbol{G}^{(a)}$ to free memory.

//Second stage: project the private gradients $\boldsymbol{G}$ into anchor subspace $\boldsymbol{B}$
Compute gradient embeddings $\boldsymbol{W} = \boldsymbol{G}\boldsymbol{B}^T$ and clip its rows with S_1 to obtain $\hat{\boldsymbol{W}}$.
Compute residual gradients $\boldsymbol{R} = \boldsymbol{G} - \boldsymbol{W}\boldsymbol{B}$ and clip its rows with S_2 to obtain $\hat{\boldsymbol{R}}$.

//Third stage: perturb gradient embedding and residual gradient separately
Perturb embedding with noise $\boldsymbol{z}^{(1)} \sim \mathcal{N}(0, \sigma_1^2\boldsymbol{I}_{k\times k})$: $\boldsymbol{w} := \sum_i \hat{\boldsymbol{W}}_{i,:}$, $\tilde{\boldsymbol{w}} := \boldsymbol{w} + \boldsymbol{z}^{(1)}$.
Perturb residual gradient with noise $\boldsymbol{z}^{(2)} \sim \mathcal{N}(0, \sigma_2^2\boldsymbol{I}_{p\times p})$: $\boldsymbol{r} := \sum_i \hat{\boldsymbol{R}}_{i,:}$, $\tilde{\boldsymbol{r}} := \boldsymbol{r} + \boldsymbol{z}^{(2)}$.
Return $\tilde{\boldsymbol{v}} := (\tilde{\boldsymbol{w}}^T\boldsymbol{B} + \tilde{\boldsymbol{r}})/n$.

A common practice to control sensitivity is to clip the output with a pre-defined threshold. In our experiments, we use different thresholds S_1 and S_2 to clip the gradient embeddings and residual gradients, respectively. The privacy loss of GEP consists of two parts: the privacy loss incurred by releasing the perturbed embedding and the privacy loss incurred by releasing the perturbed residual gradient. We compose these two parts via the Rényi differential privacy and convert it to (ϵ, δ)-DP.

We highlight several implementation techniques that make GEP widely applicable and implementable with reasonable computational cost. Firstly, auxiliary non-sensitive data do not have to be the same source as the private data and the auxiliary data can be randomly labeled. This non-sensitive data assumption is very weak and easy to satisfy in practical scenarios. To understand why random label works, a quick example is that for the least squares regression problem the individual gradient is aligned with the feature vector while the label only scales the length but does not change the direction. This auxiliary data assumption avoids conducting principal component analysis (PCA) on private gradients, which requires releasing private high-dimensional basis vectors and hence introduces large privacy loss. Secondly, we use *power method* (Panju, 2011; Vogels et al., 2019) to approximately estimate the principal components. The new operation we introduce is standard matrix multiplication that enjoys efficient implementation on GPU. The computational complexity of each power iteration is $2mkp$, where p is the number of model parameters, m is the number of anchor gradients and k is the number of subspace basis vectors. Thirdly, we divide the parameters into different groups and compute one orthonormal basis for each group. This further reduces the computational cost. For example, suppose the parameters are divided into two groups with size p_1, p_2 and the numbers of basis vectors are k_1, k_2, the computational complexity of each power iteration is $2m(k_1p_1 + k_2p_2)$, which is smaller than $2m(k_1+k_2)(p_1+p_2)$. In Appendix B, we analyze the additional computational and memory costs of GEP compared to standard gradient perturbation.

Curious readers may wonder if we can use random projection to reduce the dimensionality as Johnson–Lindenstrauss Lemma (Dasgupta & Gupta, 2003) guarantees that one can preserve the pairwise distance between any two points after projecting into a random subspace of much lower dimension. However, preserving the pairwise distance is not sufficient for high quality gradient reconstruction, which is verified by the empirical observation in Appendix C.

3.2 An analysis on the residual gradients of GEP

Let $\boldsymbol{g} := \frac{1}{n}\sum_i \boldsymbol{G}_{i,:}$ be the target private gradient. For a given anchor subspace $\boldsymbol{B}$, the residual gradients are defined as $\boldsymbol{R} := \boldsymbol{G} - \boldsymbol{G}\boldsymbol{B}^T\boldsymbol{B}$. We analyze how large the residual gradients could be. The following argument holds for all time steps and we ignore the time step index for simplicity.

For the ease of discussion, we introduce $\boldsymbol{\xi}_i := (\boldsymbol{G}_{i,:})^T$ for $i \in [n]$ to denote the the private gradients and the $\hat{\boldsymbol{\xi}}_j := (\boldsymbol{G}^{(a)}_{j,:})^T$ for $j \in [m]$ to denote the anchor gradients. We assume that the private gradients $\boldsymbol{\xi}_1, ..., \boldsymbol{\xi}_n$ and the anchor gradients $\hat{\boldsymbol{\xi}}_1, ..., \hat{\boldsymbol{\xi}}_m$ are sampled independently from a distribution $\mathcal{P}$. We assume $\boldsymbol{\Sigma} := \mathbb{E}_{\boldsymbol{\xi} \sim \mathcal{P}} \boldsymbol{\xi}\boldsymbol{\xi}^T \in \mathbb{R}^{p \times p}$ to be the population gradient (uncentered) covariance matrix. We also consider the (uncentered) empirical gradient covariance matrix $\hat{\boldsymbol{S}} := \frac{1}{m} \sum_{i=1}^m \hat{\boldsymbol{\xi}}_i \hat{\boldsymbol{\xi}}_i^T$.

One case is that the population gradient covariance matrix $\boldsymbol{\Sigma}$ is low-rank k. In this case we can argue that the residual gradients are 0 once the number of anchor gradients $m > k$.

Lemma 3.1. *Assume that the population covariance matrix $\boldsymbol{\Sigma}$ is with rank k and the distribution $\mathcal{P}$ satisfies $\mathbb{P}(\boldsymbol{\xi} \in \mathbb{F}_s) = 0$ for all s-flats $\mathbb{F}_s$ in $\mathbb{R}^p$ with $0 \leq s < k$. Let $\boldsymbol{\Sigma} = \boldsymbol{V}_k \Lambda \boldsymbol{V}_k^T$ and $\hat{\boldsymbol{S}} = \hat{\boldsymbol{V}}_{k'} \hat{\Lambda} \hat{\boldsymbol{V}}_{k'}^T$ be the eigendecompositions of $\boldsymbol{\Sigma}$ and the empirical covariance matrix $\hat{\boldsymbol{S}}$, respectively, such that $\lambda_{k'}(\hat{\boldsymbol{S}}) > 0$ and $\lambda_{k'+1}(\hat{\boldsymbol{S}}) = 0$. Then if $m \geq k$, we have with probability 1,*

$$k' = k \quad \textit{and} \quad \|\boldsymbol{V}_k \boldsymbol{V}_k^T - \hat{\boldsymbol{V}}_k \hat{\boldsymbol{V}}_k^T\|_2 = 0. \tag{2}$$

Proof. The proof is based on the non-singularity of covariance matrix. See Appendix D. □

We note that an s-flat is the translate $\mathbb{F}_s = \boldsymbol{x} + \mathbb{F}_{s(0)}$ of an s-dimensional linear subspace $\mathbb{F}_{s(0)}$ in $\mathbb{R}^p$ and the normal distribution satisfies such condition (Eaton & Perlman, 1973; Muirhead, 2009). Therefore, we have seen that for low-rank case of population covariance matrix, the residual gradients are 0 once $m > k$. In the general case, we measure the expected norm of the residual gradients.

Lemma 3.2. *Assume that $\boldsymbol{\xi} \sim \mathcal{P}$ and $\|\boldsymbol{\xi}\|^2 < T$ almost surely. Let $\boldsymbol{\Sigma} = \boldsymbol{V} \Lambda \boldsymbol{V}^T$ be the eigendecomposition of the population covariance matrix $\boldsymbol{\Sigma}$. Let $\hat{\boldsymbol{S}} = \hat{\boldsymbol{V}}_k \hat{\Lambda} \hat{\boldsymbol{V}}_k^T$ be the eigendecomposition of the empirical covariance matrix $\hat{\boldsymbol{S}}$. Then we have with probability $1 - 2\exp(-\delta)$,*

$$\mathbb{E}\|\boldsymbol{\xi} - \Pi_{\hat{\boldsymbol{V}}_k}(\boldsymbol{\xi})\|_2^2 \leq \sum_{k' > k} \lambda_{k'}(\boldsymbol{\Sigma}) + \sqrt{kC/m} + T\sqrt{2\delta/m}, \tag{3}$$

where $C = \left[\mathbb{E}\|\boldsymbol{\xi}\|^4 - \sum_i \lambda_i^2(\boldsymbol{\Sigma})\right] + \left[\frac{1}{m}\sum_{j=1}^m \|\hat{\boldsymbol{\xi}}_j\|^4 - \sum_i \lambda_i^2(\hat{\boldsymbol{S}})\right]$, $\Pi_{\hat{\boldsymbol{V}}_k}$ is a projection operator onto the subspace $\hat{\boldsymbol{V}}_k$ and the $\mathbb{E}$ is taken over the randomness of $\boldsymbol{\xi} \sim \mathcal{P}$.

Proof. The proof is an adaptation of Theorem 3.1 in Blanchard et al. (2007). □

From Lemma 3.2, we can see the larger the number of anchor gradients and the dimension of the anchor subspace k, the smaller the residual gradients. We can choose m, k properly such that the upper bound on the expected residual gradient norm is small. This indicates that we may use a smaller clipping threshold and consequently apply smaller noises with achieving the same privacy guarantee.

We next empirically examine the projection error $\boldsymbol{r} = \sum_i \boldsymbol{R}_{i,:}$ by training a 20-layer ResNet on CIFAR10 dataset. We try two different types of auxiliary data to compute the anchor gradients: 1) samples from the same source as private data with correct labels, i.e., 2000 random samples from the test data; 2) samples from different source with random labels, i.e., 2000 random samples from ImageNet. The relation between the dimension of anchor subspace k and the projection error rate ($\left\|\frac{1}{n}\boldsymbol{r}\right\| / \|\boldsymbol{g}\|$) is presented in Figure 4. We can see that the project error is small and decreases with k, and the benefit of increasing k diminishes when k is large, which is implied by Lemma 3.2. In practice one can only use small or moderate k because of the memory constraint. GEP needs to store at least k individual gradients and each individual gradient consumes the same amount of memory as the model itself. Moreover, we can see that the projection into anchor subspace of random labeled auxiliary data yields comparable projection error, corroborating our argument that unlabeled auxiliary data are sufficient for finding the anchor subspace.

We also verify that the redundancy of residual gradients is small, by plotting the stable rank ($\|\cdot\|_F / \|\cdot\|$) of residual gradient matrix in Figure 5. The stable rank of residual gradient matrix is an order of magnitude higher than the stable rank of original gradient matrix. This implies that it could be hard to further approximate $\boldsymbol{R}$ with low-dimensional embeddings.

We next compare the GEP with a scheme that simply discards the residual gradients and only outputs the perturbed gradient embedding, i.e., the output is $\tilde{\boldsymbol{u}} := \tilde{\boldsymbol{w}}^T \boldsymbol{B}/n$.

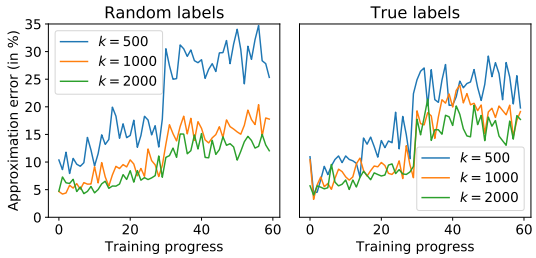

Figure 4: Relative projection error ($\left\|\frac{1}{n}\boldsymbol{r}\right\| / \|\boldsymbol{g}\|$) of the second stage in ResNet20. The number of anchor gradients is 2000. The dimension of anchor subspace is k. The learning rate is decayed by 10 at epoch 30. The left plot uses random samples from ImageNet. The right plot uses random samples from test data. The benefit of increasing k becomes smaller when k is larger.

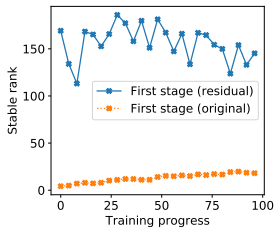

Figure 5: Stable rank of the residual gradient matrix versus original gradient matrix. The gradients are computed on full batch data for the first stage in ResNet20. The dimension of anchor subspace is $k = 1000$.

Remark 1. *Let $\tilde{\boldsymbol{u}} := \tilde{\boldsymbol{w}}^T \boldsymbol{B}/n$ be the reconstructed gradient from noisy gradient embedding and $\tilde{\boldsymbol{v}}$ be the output of GEP. If ignoring the effect of gradient clipping, we have*

$$\mathbb{E}[\tilde{\boldsymbol{u}}] = \boldsymbol{g} - \boldsymbol{r}/n, \quad \mathbb{E}[\tilde{\boldsymbol{v}}] = \boldsymbol{g}. \tag{4}$$

where $\boldsymbol{r} = \sum_i \boldsymbol{R}_{i,:}$ is the aggregated residual gradients, $\tilde{\boldsymbol{w}}, \boldsymbol{B}$ are given in Algorithm 1 and the expectation is over the added random noises.

This indicates that $\tilde{\boldsymbol{u}}$ contains a systematic error that makes $\tilde{\boldsymbol{u}}$ always deviate from $\boldsymbol{g}$ by the residual gradient. This systematic error is the projection error, which is plotted in Figure 4. The systematic error cannot be mitigated by reducing the noise magnitude (e.g., increasing the privacy budget or collecting more private data). We refer to the algorithm releasing $\tilde{\boldsymbol{u}}$ directly as *Biased-GEP* or *B-GEP* for short, which can be viewed as an efficient implementation of the algorithm in (Zhou et al., 2020). In our experiments, B-GEP can outperform standard gradient perturbation when k is large but is inferior to GEP. We note that the above remark is made with ignoring the clipping effect (or set a large clipping threshold). In practice, we do apply clipping for the individual gradients at each time step, which makes the expectations in Remark 1 obscure (Chen et al., 2020b). We note that the claim that $\tilde{\boldsymbol{v}}$ is an unbiased estimator of $\boldsymbol{g}$ is not that precise when applying gradient clipping.

3.3 Private Learning with Gradient Embedding Perturbation

GEP (Algorithm 1) describes how to release one-step gradient with privacy guarantee. In this section, we compose the privacy losses at each step to establish the privacy guarantee for the whole learning process. The differentially private learning process with GEP is given in Algorithm 2 and the privacy analysis is presented in Theorem 3.2.

Algorithm 2: Differentially private gradient descent with GEP.

1: **Input:** private dataset $\mathbb{D}$; auxiliary dataset $\mathbb{D}^{(a)}$; number of updates T; learning rate η; configuration of GEP $\mathbb{C}$; loss function ℓ;
2: **Output:** Differentially private model $\boldsymbol{\theta}_T$.
3: **for** $t = 0$ **to** $T - 1$ **do**
4: Compute the private gradients $\boldsymbol{G}_t$ and anchor gradients $\boldsymbol{G}_t^{(a)}$ of loss with respect to $\boldsymbol{\theta}_t$.
5: Call GEP with $\boldsymbol{G}_t, \boldsymbol{G}_t^{(a)}$ and configuration $\mathbb{C}$ to get $\tilde{\boldsymbol{v}}_t$.
6: Update model $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta\tilde{\boldsymbol{v}}_t$.
7: **end for**

Theorem 3.2. *For any $\epsilon < 2\log(1/\delta)$ and $\delta \in (0, 1)$, the output of Algorithm 2 satisfies (ϵ, δ)-DP if we set $\sigma \geq 2\sqrt{2T\log(1/\delta)}/\epsilon$.*

If the private gradients are randomly sampled from the full batch gradients, the privacy guarantee can be strengthened via the *privacy amplification by subsampling* theorem of DP (Balle et al., 2018; Wang et al., 2019; Zhu & Wang, 2019; Mironov et al., 2019). Theorem 3.3 gives the expected excess error of Algorithm 2. Expected excess error measures the distance between the algorithm's output and the optimal solution in expectation.

Theorem 3.3. *Suppose the loss $L(\boldsymbol{\theta}) = \frac{1}{n}\sum_{(\boldsymbol{x},y)\in\mathbb{D}} \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}), y)$ is 1-Lipschitz, convex, and β-smooth. If $\eta = \frac{1}{\beta}$, $T = \frac{n\beta\epsilon}{\sqrt{p}}$, and $\bar{\boldsymbol{\theta}} = \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{\theta}_t$, then we have*

$$\mathbb{E}[L(\bar{\boldsymbol{\theta}})] - L(\boldsymbol{\theta}_*) \leq \mathcal{O}\left(\frac{\sqrt{k\log(1/\delta)}}{n\epsilon} + \frac{\bar{r}\sqrt{p\log(1/\delta)}}{n\epsilon}\right),$$

where $\bar{r} = \frac{1}{T}\sum_{t=0}^{T-1} r_t^2$ and $r_t = \max_i \|(\boldsymbol{R}_t)_{i,:}\|$ is the sensitivity of residual gradient at step t.

The $\bar{r}$ term represents the averaged projection error over the training process. The previous best expected excess error for gradient perturbation is $\mathcal{O}(\sqrt{p\log(1/\delta)}/(n\epsilon))$ (Wang et al., 2017). As shown in Lemma 3.1, if the gradients locate in a k-dimensional subspace over the training process, $\bar{r} = 0$ and the excess error is $\mathcal{O}(\sqrt{k\log(1/\delta)}/(n\epsilon))$, independent of the problem ambient dimension p. When the gradients are in general position, i.e., gradient matrix is not exact low-rank, Lemma 3.2 and the empirical result give a hint on how small the residual gradients could be. However, it is hard to get a good bound on $\max_i \|(\boldsymbol{R}_t)_{i,:}\|$ and the bound in Theorem 3.3 does not explicitly improve over previous result. One possible solution is to use a clipping threshold based on the expected residual gradient norm. Then the output gradient becomes biased because of clipping and the utility/privacy guarantees in Theorem 3.3/3.2 require new elaborate derivation. We leave this for future work.

4 Experiments

We conduct experiments on MNIST, extended SVHN, and CIFAR-10 datasets. The model for MNIST has two convolutional layers with max-pooling and one fully connected layer. The model for SVHN and CIFAR-10 is ResNet20 in He et al. (2016). We replace all batch normalization (Ioffe & Szegedy, 2015) layers with group normalization (Wu & He, 2018) layers because batch normalization mixes the representations of different samples and makes the privacy loss cannot be analyzed accurately. The non-private accuracy for MNIST, SVHN, and CIFAR-10 is 99.1%, 95.9%, and 90.4%, respectively.

We also provide experiments with pre-trained models in Appendix A. Notably, with a model pre-trained on unlabeled ImageNet, GEP achieves 94.8% validation accuracy on CIFAR10 with a privacy bound of $\epsilon = 2$.

Evaluated algorithms We use the algorithm in Abadi et al. (2016) as benchmark gradient perturbation approach, referred to as "GP". We also compare GEP with PATE (Papernot et al., 2017). The privacy parameter ϵ of PATE is data-dependent and hence cannot be released directly (see Section 3.3 in Papernot et al. (2017)). Nonetheless, we report the results of PATE anyway.

Implementation details At each step, GEP needs to release two vectors: the noisy gradient embedding and the noisy residual gradient. The gradient embeddings have a sensitivity of S_1 and the residual gradients have a sensitivity of S_2 because of the clipping. The output of GEP can be constructed as follows: (1) normalize the gradient embeddings and residual gradients by $1/S_1$ and $1/S_2$, respectively, (2) concatenate the rescaled vectors, (3) release the concatenated vector via gaussian mechanism with sensitivity $\sqrt{2}$, (4) rescale the two components by S_1 and S_2. B-GEP only needs to release the normalized noisy gradient embedding. We use the numerical tool in Mironov et al. (2019) to compute the privacy loss. For given privacy budget and sampling probability, σ is set to be the smallest value such that the privacy budget is allowable to run desired epochs.

All experiments are run on a single Tesla V100 GPU with 16G memory. For ResNet20, the parameters are divided into five groups: input layer, output layer, and three intermediate stages. For a given quota of basis vectors, we allocate it to each group according to the square root of the number of parameters in each group. We compute an orthonormal subspace basis on each group separately. Then we concatenate the projections of all groups to construct gradient embeddings. The number of power iterations t is set as 1 as empirical evaluations suggest more iterations do not improve the performance for GEP and B-GEP.

Table 1: Test performance (in %) with DP guarantee. For PATE, we use the numbers reported in Papernot et al. (2018). Symbol Δ denotes the improvement over GP baseline.

Dataset	Algorithm	$\epsilon = 1.99$	Δ	$\epsilon = 4.97$	Δ	$\epsilon = 7.98$	Δ
MNIST	GP	94.7	+0.0	96.8	+0.0	97.2	+0.0
	PATE	98.5	**+3.8**	98.5	**+1.7**	98.5	**+1.3**
	B-GEP	93.1	-1.6	94.5	-2.3	95.9	-1.3
	GEP	96.3	+1.6	97.9	**+1.1**	98.4	**+1.2**
SVHN	GP	87.1	+0.0	91.3	+0.0	91.6	+0.0
	PATE	/	/	91.6	+0.3	91.6	+0.0
	B-GEP	88.5	+1.4	91.8	+0.5	92.3	+0.7
	GEP	92.3	**+5.2**	94.7	**+3.4**	95.1	**+3.5**
CIFAR-10	GP	43.6	+0.0	52.2	+0.0	56.4	+0.0
	B-GEP	50.3	+6.7	59.5	+7.3	63.0	+6.6
	GEP	59.7	**+16.1**	70.1	**+17.9**	74.9	**+18.5**

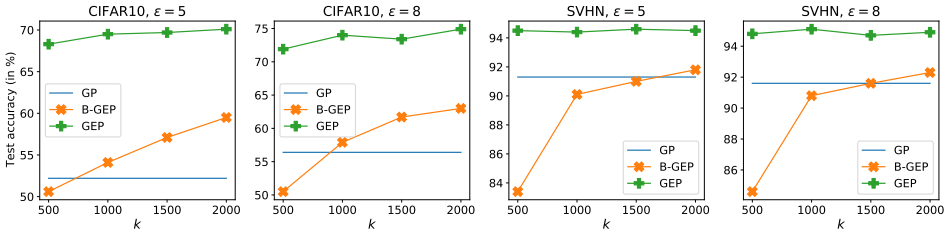

Figure 6: Test accuracy when varying the dimension of anchor subspace. GEP significantly outperforms B-GEP for all k. Moreover, the performance of GEP is not that sensitive to k.

For all datasets, the anchor gradients are computed on 2000 random samples from ImageNet. In Appendix C, we examine the influence of choosing different numbers of anchor gradients and different sources of auxiliary data. The selected images are downsampled into size of 32×32 (28×28 for MNIST) and we label them randomly at each update. For SVHN and CIFAR-10, k is chosen from $[500, 1000, 1500, 2000]$. For MNIST, we halve the size of k. We use SGD with momentum 0.9 as the optimizer. Initial learning rate and batchsize are 0.1 and 1000, respectively. The learning rate is divided by 10 at middle of training. Weight decay is set as 1×10^{-4}. The clipping threshold for is 10 for original gradients and 2 for residual gradients. The number of training epochs for CIFAR-10 and MNIST is 50, 100, 200 for privacy parameter $\epsilon = 2, 5, 8$, respectively. The number of training epochs for SVHN is 5, 10, 20 for privacy parameter $\epsilon = 2, 5, 8$, respectively. Privacy parameter δ is 10^{-6} for SVHN and 10^{-5} for CIFAR-10 and MNIST.

Results The best accuracy with given ϵ is in Table 4. For all datasets, GEP achieves considerable improvement over GP in Abadi et al. (2016). Specifically, GEP achieves 74.9% test accuracy on CIFAR-10 with $(8, 10^{-5})$-DP, outperforming GP by 18.5%. PATE achieves best accuracy on MNIST but its performance drops as the dataset becomes more complex.

We also plot the relation between accuracy and k in Figure 6. GEP is less sensitive to the choice of k and outperforms B-GEP for all choices of k. The improvement of increasing k becomes smaller as k becomes larger. We note that the memory cost of choosing large k is high because we need to store at least k individual gradients to compute anchor subspace.

5 Conclusion

In this paper, we propose Gradient Embedding Perturbation (GEP) for learning with differential privacy. GEP leverages the gradient redundancy to reduce the added noise and outputs an unbiased estimator of target gradient. The several key designs of GEP significantly boost the applicability of GEP. Extensive experiments on real world datasets demonstrate the superior utility of GEP. In the future, it is exciting to explore the usage of GEP in other kinds of tasks under the constraint of differential privacy, e.g., language modeling (McMahan et al., 2018) and GAN (Xie et al., 2018).

REFERENCES

Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *ACM SIGSAC Conference on Computer and Communications Security*, 2016.

Noga Alon, Raef Bassily, and Shay Moran. Limits of private learning with access to public data. In *Advances in Neural Information Processing Systems*, 2019.

Anonymous. Differentially private learning needs better features (or much more data). *OpenReview*, 2020. URL `https://openreview.net/forum?id=YTWGvpFOQD-`.

Borja Balle, Gilles Barthe, and Marco Gaboardi. Privacy amplification by subsampling: Tight analyses via couplings and divergences. In *Advances in Neural Information Processing Systems*, 2018.

Raef Bassily, Adam Smith, and Abhradeep Thakurta. Differentially private empirical risk minimization: Efficient algorithms and tight error bounds. *Annual Symposium on Foundations of Computer Science*, 2014.

Daniel Bernau, Philip-William Grassal, Jonas Robl, and Florian Kerschbaum. Assessing differentially private deep learning with membership inference. *arXiv preprint arXiv:1912.11328*, 2019.

Gilles Blanchard, Olivier Bousquet, and Laurent Zwald. Statistical properties of kernel principal component analysis. *Machine Learning*, 66(2-3):259–294, 2007.

Mark Bun and Thomas Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography Conference*, 2016.

Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *USENIX Security Symposium*, 2019.

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. *arXiv preprint arXiv:2006.10029*, 2020a.

Xiangyi Chen, Steven Z Wu, and Mingyi Hong. Understanding gradient clipping in private sgd: A geometric perspective. *Advances in Neural Information Processing Systems*, 33, 2020b.

Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of johnson and lindenstrauss. *Random Structures & Algorithms*, 2003.

Jinshuo Dong, Aaron Roth, and Weijie J Su. Gaussian differential privacy. *arXiv preprint arXiv:1905.02383*, 2019.

Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2006a.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, 2006b.

Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 2014.

Morris L Eaton and Michael D Perlman. The non-singularity of generalized sample covariance matrices. *The Annals of Statistics*, pp. 710–717, 1973.

Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *ACM SIGSAC Conference on Computer and Communications Security*, 2015.

Kazuto Fukuchi, Quang Khai Tran, and Jun Sakuma. Differentially private empirical risk minimization with input perturbation. In *International Conference on Discovery Science*, 2017.

Mary Gooneratne, Khe Chai Sim, Petr Zadrazil, Andreas Kabel, Françoise Beaufays, and Giovanni Motta. Low-rank gradient approximation for memory-efficient on-device training of deep neural network. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020.

Guy Gur-Ari, Daniel A Roberts, and Ethan Dyer. Gradient descent happens in a tiny subspace. *arXiv preprint arXiv:1812.04754*, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Conference on Computer Vision and Pattern Recognition*, 2020.

Briland Hitaj, Giuseppe Ateniese, and Fernando Pérez-Cruz. Deep models under the gan: information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, 2015.

Roger Iyengar, Joseph P Near, Dawn Song, Om Thakkar, Abhradeep Thakurta, and Lun Wang. Towards practical differentially private convex optimization. In *IEEE Symposium on Security and Privacy*, 2019.

Bargav Jayaraman, Lingxiao Wang, David Evans, and Quanquan Gu. Distributed learning without distress: Privacy-preserving empirical risk minimization. In *Advances in Neural Information Processing Systems*, 2018.

James Jordon, Jinsung Yoon, and Mihaela van der Schaar. Pate-gan: Generating synthetic data with differential privacy guarantees. In *International Conference on Learning Representations*, 2019.

Peter Kairouz, Mónica Ribero, Keith Rush, and Abhradeep Thakurta. Dimension independence in unconstrained private erm via adaptive preconditioning. *arXiv preprint arXiv:2008.06570*, 2020.

Daniel Kifer, Adam Smith, and Abhradeep Thakurta. Private convex empirical risk minimization and high-dimensional regression. In *Conference on Learning Theory*, 2012.

Jaewoo Lee and Daniel Kifer. Concentrated differentially private gradient descent with adaptive per-iteration privacy budget. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018.

Xinyan Li, Qilong Gu, Yingxue Zhou, Tiancong Chen, and Arindam Banerjee. Hessian based analysis of sgd for deep nets: Dynamics and generalization. In *SIAM International Conference on Data Mining*, 2020.

Yuzhe Ma, Xiaojin Zhu, and Justin Hsu. Data poisoning against differentially-private learners: attacks and defenses. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pp. 4732–4738. AAAI Press, 2019.

H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. In *International Conference on Learning Representations*, 2018.

Ilya Mironov. Rényi differential privacy. In *IEEE Computer Security Foundations Symposium*, 2017.

Ilya Mironov, Kunal Talwar, and Li Zhang. Rényi differential privacy of the sampled gaussian mechanism. *arXiv*, 2019.

Robb J Muirhead. *Aspects of multivariate statistical theory*, volume 197. John Wiley & Sons, 2009.

Maysum Panju. Iterative methods for computing eigenvalues and eigenvectors. *arXiv preprint arXiv:1105.1185*, 2011.

Nicolas Papernot, Martín Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. 2017.

Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. Scalable private learning with pate. 2018.

NhatHai Phan, My T Thai, Han Hu, Ruoming Jin, Tong Sun, and Dejing Dou. Scalable differential privacy with certified robustness in adversarial learning. *International Conference on Machine Learning*, 2020.

Md Atiqur Rahman, Tanzila Rahman, Robert Laganiere, Noman Mohammed, and Yang Wang. Membership inference attack against differentially private deep learning model. *Transactions on Data Privacy*, 2018.

Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *ACM SIGSAC conference on computer and communications security*, 2015.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *IEEE Symposium on Security and Privacy (SP)*, 2017.

Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. Stochastic gradient descent with differentially private updates. In *Global Conference on Signal and Information Processing (GlobalSIP)*, 2013.

Kunal Talwar, Abhradeep Guha Thakurta, and Li Zhang. Nearly optimal private lasso. In *Advances in Neural Information Processing Systems*, 2015.

Abhradeep Guha Thakurta and Adam Smith. Differentially private feature selection via stability arguments, and the robustness of the lasso. In *Conference on Learning Theory*, 2013.

Thijs Vogels, Sai Praneeth Karimireddy, and Martin Jaggi. Powersgd: Practical low-rank gradient compression for distributed optimization. In *Advances in Neural Information Processing Systems*, 2019.

Di Wang and Jinhui Xu. On sparse linear regression in the local differential privacy model. In *International Conference on Machine Learning*, 2019.

Di Wang, Minwei Ye, and Jinhui Xu. Differentially private empirical risk minimization revisited: Faster and more general. In *Advances in Neural Information Processing Systems*, 2017.

Jun Wang and Zhi-Hua Zhou. Differentially private learning with small public data. In *AAAI*, 2020.

Lingxiao Wang and Quanquan Gu. Differentially private iterative gradient hard thresholding for sparse learning. In *International Joint Conference on Artificial Intelligence*, 2019.

Yu-Xiang Wang, Borja Balle, and Shiva Prasad Kasiviswanathan. Subsampled rényi differential privacy and analytical moments accountant. In *International Conference on Artificial Intelligence and Statistics*, 2019.

Xi Wu, Matthew Fredrikson, Somesh Jha, and Jeffrey F Naughton. A methodology for formalizing model-inversion attacks. In *IEEE Computer Security Foundations Symposium*, 2016.

Xi Wu, Fengan Li, Arun Kumar, Kamalika Chaudhuri, Somesh Jha, and Jeffrey Naughton. Bolt-on differential privacy for scalable stochastic gradient descent-based analytics. In *ACM International Conference on Management of Data*, 2017.

Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, 2018.

Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. Differentially private generative adversarial network. *arXiv preprint arXiv:1802.06739*, 2018.

Yingxue Zhou, Zhiwei Steven Wu, and Arindam Banerjee. Bypassing the ambient dimension: Private sgd with gradient subspace identification. *arXiv preprint arXiv:2007.03813*, 2020.

Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In *Advances in Neural Information Processing Systems*, 2019.

Yuqing Zhu and Yu-Xiang Wang. Poission subsampled rényi differential privacy. In *International Conference on Machine Learning*, 2019.

A EXPERIMENTS WITH PRE-TRAINED MODELS

Recent works have shown that pre-training the models on unlabeled data can be beneficial for subsequent learning tasks (Chen et al., 2020a; He et al., 2020). Anonymous (2020) demonstrate that differentially private linear classifier can achieve high accuracy using the features produced by those per-trained models. We show that GEP can also benefit from such pre-trained models.

Inspired by Anonymous (2020), we use the output of the penultimate layer of a pre-trained ResNet152 model as feature to train a private linear classifier. The ResNet152 model is pre-trained on unlabeled ImageNet using SimCLR (Chen et al., 2020a). The feature dimension is 4096.

Implementation Details We choose the privacy parameter ϵ from $[0.1, 0.5, 1, 2]$. The privacy parameter δ is 1×10^{-5}. We run all experiments for 5 times and report the average accuracy. The clipping threshold of residual gradients is still one-fifth of the clipping threshold of the original gradients. The dimension of anchor subspace is set as $200 \sim \sqrt{p}$ where $p = 40960$ is the model dimension. The optimizer is Adam with default momentum coefficients. Other hyper-parameters are listed in Table 2.

Hyperparameter	Values
Learning rate	0.01, 0.05, 0.1
Running steps	50, 100, 400
Clipping threshold	0.01, 0.1, 1

Table 2: Hyperparameter values used in Appendix A.

Results The experiment results are shown in Table 3. GEP outperforms GP on all values of ϵ. With privacy bound $\epsilon = 2$, GEP achieves 94.8% validation accuracy on CIFAR10 dataset, improving over the GP baseline by 1.4%. For very strong privacy guarantee ($\epsilon = 0.1$), B-GEP performs on par with GEP because strong privacy guarantee requires large noise and the useful signal in residual gradient is submerged in the added noise. B-GEP benefits less from larger ϵ compared to GP or GEP. For $\epsilon = 1$ and 2, the performance of B-GEP is worse than the performance of GP. This is because larger ϵ can not reduce the systematic error of B-GEP (see Remark 1 in Section 3.2).

Table 3: Validation accuracy (in %) on CIFAR10 with varying choices of ϵ. We train a private linear model on top of the features from a ResNet152 model, which is pre-trained on unlabeled ImageNet.

	$\epsilon = 0.1$	$\epsilon = 0.5$	$\epsilon = 1$	$\epsilon = 2$
Non private	96.3	96.3	96.3	96.3
GP	88.2 ($\pm$0.16)	91.1 ($\pm$0.17)	93.2 ($\pm$0.19)	93.4 ($\pm$0.12)
B-GEP	**91.0** ($\pm$0.07)	92.9 ($\pm$0.03)	93.1 ($\pm$0.10)	93.2 ($\pm$0.08)
GEP	90.9 ($\pm$0.19)	**93.5** ($\pm$0.06)	**94.3** ($\pm$0.09)	**94.8** ($\pm$0.06)

B COMPLEXITY ANALYSIS

We provide an analysis of the computational and memory costs of the construction of anchor subspace. The computation of the anchor subspace is the dominant additional cost of GEP compared to conventional gradient perturbation. Notations: k, m, n, and p are the dimension of anchor subspace, number of anchor gradients, number of private gradients, and the model dimension, respectively. In order to reduce the computational and memory costs, we divide the parameters into g groups and compute one orthonormal basis for each group. We refer to this approach as 'parameter grouping'. In this section, we assume the parameters and the dimension of the anchor subspace are both divided evenly. Table 4 summarizes the additional costs of GEP with/without parameter grouping. Using parameter grouping can reduce the computational/memory cost significantly.

C ABLATION STUDY

The influence of choosing different auxiliary datasets. We conduct experiments with different choices of auxiliary datasets. For CIFAR10, we try 2000 random test samples from CIFAR10, 2000

Table 4: Computational and memory costs of a single power iteration in Algorithm 1. The computation cost is measured by the number of floating point operations. The memory cost is measured by the number of floating-point numbers we need to store. 'GEP+PG' denotes GEP with parameter grouping and g denotes the number of groups. Notations: k, m, n, and p are the dimension of anchor subspace, number of anchor gradients, number of private gradients, and the model dimension, respectively.

	Computational Cost	Memory Cost
GEP	$2mkp$	$\max(0, (m+k-n)p)$
GEP+PG	$2mkp/g$	$\max(0, (m+k/g-n)p)$

random samples from CIFAR100, and 2000 random samples from ImageNet. When the auxiliary dataset is CIFAR10, we try both correct labels and random labels. For all choices of auxiliary datasets, the test accuracy is evaluated on 8000 test samples of CIFAR10 that are not used as auxiliary data. Other implementation details are the same as in Section 4. The results are shown in Table 5. Surprisingly, using samples from CIFAR10 with correct labels yields the worst accuracy. This may because the model 'overfits' the auxiliary data when it has access to correct labels, which makes the anchor subspace contains less information about the private gradients. The best accuracy is achieved using samples from CIFAR10 with random labels, this makes sense because in this case the features of auxiliary data and private data have the same distribution. Using samples from CIFAR100 or ImageNet as auxiliary data has a small influence on the test accuracy.

Table 5: Test accuracy on CIFAR10 with different choices of auxiliary datasets. The privacy guarantee is $(8, 10^{-5})$-DP. We report the average accuracy of five runs with standard deviations in brackets.

Auxiliary Data	Random Label?	Test Accuracy
CIFAR10	No	72.9 ($\pm$0.31)
CIFAR10	Yes	**75.1** ($\pm$0.42)
CIFAR100	Yes	74.7 ($\pm$0.46)
ImageNet	Yes	74.8 ($\pm$0.39)

The influence of the number of anchor gradients. In the main text, the size of auxiliary dataset is $m = 2000$. We conduct more experiments with different sizes of auxiliary dataset to examine the influence of m. The auxiliary data is randomly sampled from ImageNet. Table 6 reports the test accuracy on CIFAR10 with different choices of m. For both B-GEP and GEP, increasing m leads to slightly improved performance.

Table 6: Test accuracy on CIFAR10 with different sizes of auxiliary dataset. The privacy guarantee is $(8, 10^{-5})$-DP. We report the average accuracy of five runs with standard deviations in brackets.

Algorithm	$m = 1000$	$m = 2000$	$m = 4000$
B-GEP	62.2 ($\pm$0.26)	62.6 ($\pm$0.24)	63.3 ($\pm$0.27)
GEP	74.6 ($\pm$0.41)	74.8 ($\pm$0.39)	**75.2** ($\pm$0.34)

The projection error of random basis vectors. It is tempting to construct the anchor subspace using random basis vectors because Johnson–Lindenstrauss Lemma (Dasgupta & Gupta, 2003) guarantees that one can preserve the pairwise distance between any two points after projecting into a random subspace of much lower dimension. We empirically verify the projection error of Gaussian random basis vectors on CIFAR10 and SVHN. The experiment settings are the same as in Section 4. The projection errors over the training process are plotted in Figure 7. The projection error of random basis vectors is very high ($> 95\%$) throughout training. This is because preserving the pairwise distance is not sufficient for high quality gradient reconstruction, which requires one to preserve the average 'distance' between any individual gradient and all other gradients.

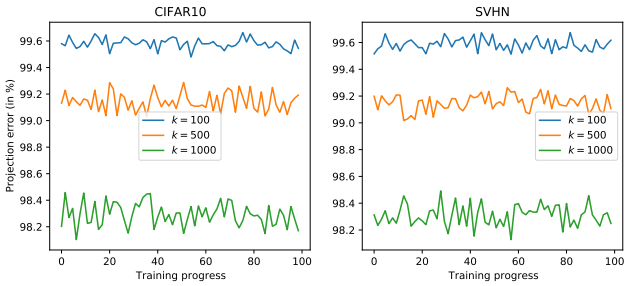

Figure 7: Projection error rate of random basis vectors. The dimension of subspace is denoted by k.

D Missing Proofs

Lemma 3.1. *Assume that the population covariance matrix $\boldsymbol{\Sigma}$ is with rank k and the distribution $\mathcal{P}$ satisfies $\mathbb{P}(\boldsymbol{\xi} \in \mathbb{F}_s) = 0$ for all s-flats $\mathbb{F}_s$ in $\mathbb{R}^p$ with $0 \leq s < k$. Let $\boldsymbol{\Sigma} = \boldsymbol{V}_k \Lambda \boldsymbol{V}_k^T$ and $\hat{\boldsymbol{S}} = \hat{\boldsymbol{V}}_{k'} \hat{\Lambda} \hat{\boldsymbol{V}}_{k'}^T$ be the eigendecompositions of $\boldsymbol{\Sigma}$ and the empirical covariance matrix $\hat{\boldsymbol{S}}$, respectively, such that $\lambda_{k'}(\hat{\boldsymbol{S}}) > 0$ and $\lambda_{k'+1}(\hat{\boldsymbol{S}}) = 0$. Then if $m \geq k$, we have with probability 1,*

$$k' = k \quad \textit{and} \quad \|\boldsymbol{V}_k \boldsymbol{V}_k^T - \hat{\boldsymbol{V}}_k \hat{\boldsymbol{V}}_k^T\|_2 = 0. \tag{2}$$

Proof. We extend the Theorem 3.2 in Eaton & Perlman (1973) to the low-rank case.

Theorem D.1 (Theorem 3.2 in Eaton & Perlman (1973))**.** *Let $\boldsymbol{X} = (\boldsymbol{x}_1, ..., \boldsymbol{x}_n)$ where the $\boldsymbol{x}_i$ are i.i.d. random vectors in $\mathbb{R}^p$, $n \geq p$. If $\mathbb{P}\{\boldsymbol{x}_1 \in \mathbb{M}\} = 0$ for all proper manifolds $\mathbb{M} \subset \mathbb{R}^p$, then $\mathbb{P}\{\boldsymbol{X}$ is non-singular$\}$=1.*

We note that the subspace spanned by $\hat{\boldsymbol{V}}_{k'}$ is in the space spanned by $\boldsymbol{V}_k$ by definition. Hence $k' \leq k$.

Let $\hat{\boldsymbol{x}}_i := \boldsymbol{V}_k^T \hat{\boldsymbol{\xi}}_i \in \mathbb{R}^k$ for $i \in [m]$. Then $\hat{\boldsymbol{X}} := (\hat{\boldsymbol{x}}_1, ..., \hat{\boldsymbol{x}}_m)$ is non-singular because of the assumption and Theorem D.1. That is $rank(\hat{\boldsymbol{X}}) = k$. Therefore $rank((\hat{\boldsymbol{\xi}}_1, ..., \hat{\boldsymbol{\xi}}_m)) \geq k$, $rank(\hat{\boldsymbol{S}}) \geq k$ and $k' \geq k$. Therefore $k' = k$ and the subspace spanned by $\hat{\boldsymbol{V}}_{k'}$ and the subspace spanned by $\boldsymbol{V}_k$ are identical. □

Theorem 3.1. *Let S_1 and S_2 be the sensitivity of $\boldsymbol{w}$ and $\boldsymbol{r}$, respectively, the output of Algorithm 1 satisfies (ϵ, δ)-DP for any $\delta \in (0, 1)$ and $\epsilon \leq 2\log(1/\delta)$ if we choose $\sigma_1 \geq 2S_1\sqrt{2\log(1/\delta)}/\epsilon$ and $\sigma_2 \geq 2S_2\sqrt{2\log(1/\delta)}/\epsilon$.*

Proof of Theorem 3.1. We first introduce some background knowledge of Rényi differential privacy (RDP) (Mironov, 2017). RDP measures the Rényi divergence between two output distributions.

Definition 2 ((λ, γ)-RDP)**.** *A randomized mechanism f is said to guarantee (λ, γ)-RDP if for any neighboring datasets $\mathbb{D}, \mathbb{D}'$ and $\lambda > 1$ it holds that*

$$D_\lambda(f(\mathbb{D})||f(\mathbb{D}')) \leq \gamma,$$

where $D_\lambda(\cdot||\cdot)$ denotes the Rényi divergence of order λ.

We next introduce some useful properties of RDP.

Lemma D.2 (Gaussian mechanism of RDP)**.** *Let $S = \max_{\mathbb{D}\sim\mathbb{D}'} \|f(\mathbb{D}) - f(\mathbb{D}')\|$ be the l_2 sensitivity, then Gaussian mechanism $\mathcal{M} = f(\mathbb{D}) + \boldsymbol{z}$ satisfies $(\lambda, \frac{\lambda S^2}{2\sigma^2})$-RDP, where $\boldsymbol{z} \sim \mathcal{N}(0, \sigma^2 I_{p\times p})$.*

Lemma D.3 (Composition of RDP)**.** *If M_1, M_2 satisfy (λ, γ_1)-RDP and (λ, γ_2)-RDP respectively, then their composition satisfies $(\lambda, \gamma_1 + \gamma_2)$-RDP.*

Lemma D.4 (Conversion from RDP to (ϵ, δ)-DP)**.** *If $\mathcal{M}$ obeys (λ, γ)-RDP, then $\mathcal{M}$ obeys $(\gamma + \log(1/\delta)/(\lambda - 1), \delta)$-DP for all $0 < \delta < 1$.*

Now we proof Theorem 3.1. Let $\boldsymbol{W}, \boldsymbol{W}'$ be the gradient embeddings of two neighboring datasets $\mathbb{D} \sim \mathbb{D}'$ and $\boldsymbol{R}, \boldsymbol{R}'$ be corresponding residual gradients. Without loss of generality, suppose $\boldsymbol{W}$ ($\boldsymbol{R}$) has one more row than $\boldsymbol{W}'$ ($\boldsymbol{R}'$). For given sensitivity S_1, S_2,

$$\max_{\mathbb{D}\sim\mathbb{D}'} \|\boldsymbol{w} - \boldsymbol{w}'\| = \max_{\boldsymbol{W}\sim\boldsymbol{W}'} \|\boldsymbol{W}_{n,:}\| \leq S_1, \quad \max_{\mathbb{D}\sim\mathbb{D}'} \|\boldsymbol{r} - \boldsymbol{r}'\| = \max_{\boldsymbol{R}\sim\boldsymbol{R}'} \|\boldsymbol{R}_{n,:}\| \leq S_2.$$

If we set $\sigma_1 = S_1\sigma$ and $\sigma_2 = S_2\sigma$ for some σ, then Algorithm 1 satisfies $(\lambda, \frac{\lambda}{\sigma^2})$-RDP because of Lemma D.2 and D.3. In order to guarantee (ϵ, δ)-DP, we need

$$\frac{\lambda}{\sigma^2} + \frac{\log(1/\delta)}{\lambda - 1} \leq \epsilon. \tag{5}$$

Choose $\lambda = 1 + \frac{2\log(1/\delta)}{\epsilon}$ and rearrange Eq (5), we need

$$\sigma^2 \geq \frac{2\left(\epsilon + 2\log(1/\delta)\right)}{\epsilon^2}. \tag{6}$$

Then using the constraint on ϵ concludes the proof.

□

Theorem 3.2. *For any $\epsilon < 2\log(1/\delta)$ and $\delta \in (0, 1)$, the output of Algorithm 2 satisfies (ϵ, δ)-DP if we set $\sigma \geq 2\sqrt{2T\log(1/\delta)}/\epsilon$.*

Proof of Theorem 3.2. From the proof of Theorem 3.1, we have each call of GEP satisfies $(\lambda, \frac{\lambda}{\sigma^2})$-RDP. Then by the composition property of RDP (Lemma D.3), the output of Algorithm 2 satisfies $(\lambda, \frac{T\lambda}{\sigma^2})$-RDP. Plugging $\frac{T\lambda}{\sigma^2}$ into Equation 5 and 6 concludes the proof.

□

Theorem 3.3. *Suppose the loss $L(\boldsymbol{\theta}) = \frac{1}{n}\sum_{(\boldsymbol{x},y)\in\mathbb{D}} \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}), y)$ is 1-Lipschitz, convex, and β-smooth. If $\eta = \frac{1}{\beta}$, $T = \frac{n\beta\epsilon}{\sqrt{p}}$, and $\bar{\boldsymbol{\theta}} = \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{\theta}_t$, then we have*

$$\mathbb{E}[L(\bar{\boldsymbol{\theta}})] - L(\boldsymbol{\theta}_*) \leq \mathcal{O}\left(\frac{\sqrt{k\log(1/\delta)}}{n\epsilon} + \frac{\bar{r}\sqrt{p\log(1/\delta)}}{n\epsilon}\right),$$

where $\bar{r} = \frac{1}{T}\sum_{t=0}^{T-1} r_t^2$ and $r_t = \max_i \|(\boldsymbol{R}_t)_{i,:}\|$ is the sensitivity of residual gradient at step t.

Proof of Theorem 3.3. The β-smooth condition gives

$$L(\boldsymbol{\theta}_{t+1}) \leq L(\boldsymbol{\theta}_t) + \langle \nabla L(\boldsymbol{\theta}_t), \boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t \rangle + \frac{\beta}{2}\|\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t\|^2. \tag{7}$$

Based on the update rule of GEP we have

$$\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t = -\eta\tilde{\boldsymbol{v}} = -\eta\nabla L(\boldsymbol{\theta}_t) - \frac{\eta}{n}(\boldsymbol{z}_t^{(1)}\boldsymbol{B} + \boldsymbol{z}_t^{(2)}), \tag{8}$$

where $\boldsymbol{z}_t^{(1)} \sim \mathcal{N}(0, \sigma^2\boldsymbol{I}_{k\times k})$, $\boldsymbol{z}_t^{(2)} \sim \mathcal{N}(0, \sigma^2 r_t^2 \boldsymbol{I}_{p\times p})$ are the perturbation noises and $r_t = \max_i \|(\boldsymbol{R}_t)_{i,:}\|$ is the sensitivity of residual gradients at step t.

Take expectation on Eq (7) with respect to the perturbation noises.

$$\mathbb{E}[L(\boldsymbol{\theta}_{t+1})] \leq \mathbb{E}[L(\boldsymbol{\theta}_t)] - (\eta - \beta\eta^2/2)\mathbb{E}[\|\nabla L(\boldsymbol{\theta}_t)\|^2] + \frac{\beta\eta^2\sigma^2}{2n^2}\left(k + pr_t^2\right). \tag{9}$$

Subtract $L(\boldsymbol{\theta}_*)$ from both sides, we have

$$\begin{aligned}\mathbb{E}[L(\boldsymbol{\theta}_{t+1})] - L(\boldsymbol{\theta}_*) &\leq \mathbb{E}[L(\boldsymbol{\theta}_t)] - L(\boldsymbol{\theta}_*) - (\eta - \beta\eta^2/2)\mathbb{E}[\|\nabla L(\boldsymbol{\theta}_t)\|^2] + \frac{\beta\eta^2\sigma^2}{2n^2}\left(k + pr_t^2\right) \\ &\leq \mathbb{E}[\langle \nabla L(\boldsymbol{\theta}_t), \boldsymbol{\theta}_t - \boldsymbol{\theta}_* \rangle] - (\eta - \beta\eta^2/2)\mathbb{E}[\|\nabla L(\boldsymbol{\theta}_t)\|^2] + \frac{\beta\eta^2\sigma^2}{2n^2}\left(k + pr_t^2\right).\end{aligned} \tag{10}$$

The second inequality holds because L is convex. Then choose $\eta = \frac{1}{\beta}$ and plug $\nabla L(\boldsymbol{\theta}_t) = (\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t+1})/\eta - (\boldsymbol{z}_1^t \boldsymbol{B} + \boldsymbol{z}_2^t)/n$ into Eq (10).

$$\begin{aligned} \mathbb{E}[L(\boldsymbol{\theta}_{t+1})] - L(\boldsymbol{\theta}_*) &\leq \beta \mathbb{E}[\langle \boldsymbol{\theta}_t - \boldsymbol{\theta}_{t+1}, \boldsymbol{\theta}_t - \boldsymbol{\theta}_* \rangle] - \frac{\beta}{2} \mathbb{E}[\|\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t+1}\|^2] + \frac{\sigma^2}{\beta n^2} \left(k + p r_t^2\right) \\ &= \frac{\beta}{2} \left(\mathbb{E}[\|\boldsymbol{\theta}_t - \boldsymbol{\theta}_*\|^2] - \mathbb{E}[\|\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_*\|^2] \right) + \frac{\sigma^2}{\beta n^2} \left(k + p r_t^2\right). \end{aligned} \tag{11}$$

Sum over $t = 0, \ldots, T-1$ and use convexity, we have

$$\mathbb{E}[L(\bar{\boldsymbol{\theta}})] - L(\boldsymbol{\theta}_*) \leq \frac{\beta}{2T} \|\boldsymbol{\theta}_0 - \boldsymbol{\theta}_*\| + \frac{\sigma^2}{\beta n^2} (k + \frac{p}{T} \sum_{t=0}^{T-1} r_t^2). \tag{12}$$

Then substituting $T = \frac{n\beta\epsilon}{\sqrt{p}}$ and $\sigma = \mathcal{O}(\sqrt{T \log(1/\delta)}/\epsilon)$ yields the desired bound.

□