Private Adaptive Optimization with Side information

Tian Li [1] **Manzil Zaheer** [2] **Sashank J. Reddi** [3] **Virginia Smith** [1]

Abstract

Adaptive optimization methods have become the default solvers for many machine learning tasks. Unfortunately, the benefits of adaptivity may degrade when training with differential privacy, as the noise added to ensure privacy reduces the effectiveness of the adaptive preconditioner. To this end, we propose `AdaDPS`, a general framework that uses *non-sensitive side information* to precondition the gradients, allowing the effective use of adaptive methods in private settings. We formally show `AdaDPS` reduces the amount of noise needed to achieve similar privacy guarantees, thereby improving optimization performance. Empirically, we leverage simple and readily available side information to explore the performance of `AdaDPS` in practice, comparing to strong baselines in both centralized and federated settings. Our results show that `AdaDPS` improves accuracy by 7.7% (absolute) on average—yielding state-of-the-art privacy-utility trade-offs on large-scale text and image benchmarks.

1. Introduction

Privacy-sensitive applications in areas such as healthcare and cross-device federated learning have fueled a demand for optimization methods that ensure *differential privacy (DP)* (Dwork et al., 2006; Chaudhuri et al., 2011; Abadi et al., 2016; McMahan et al., 2018). These methods typically perturb gradients with random noise at each iteration in order to mask the influence of individual examples on the trained model. As the amount of privacy is directly related to the number of training iterations, private applications stand to benefit from optimizers that improve convergence speed. To capitalize on this, a number of recent works have naturally tried to combine DP with adaptive optimizers such as Adagrad, RMSProp, and Adam, which have proven to be effective for non-private machine learning tasks, especially those involving sparse gradients or non-uniform stochastic noise (Duchi et al., 2011; Hinton et al., 2012; Kingma & Ba, 2015; Reddi et al., 2018a;b; Zhang et al., 2020).

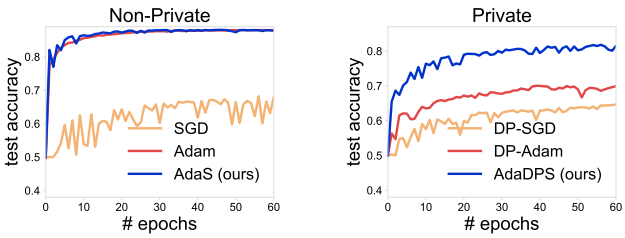

Figure 1. Test performance on IMDB with logistic regression. `AdaS` refers to using preconditioning in `AdaDPS` for non-private training. Adaptive methods (Adam) become less effective when trained with differential privacy (DP-Adam), while `AdaDPS` retains the benefits of adaptivity.

Unfortunately, tasks where adaptive optimizers work particularly well (e.g., sparse, high-dimensional problems), are exactly the tasks where DP is known to degrade performance (Bassily et al., 2014). Indeed, as we show in Figure 1, this can result in existing private adaptive optimization methods performing *only marginally better* than simple baselines such as differentially private stochastic gradient descent (DP-SGD), even under generous privacy budgets.

In this work, we aim to close the gap between adaptive optimization in non-private and private settings. We propose `AdaDPS`, a simple yet powerful framework leveraging non-sensitive side information to effectively adapt to the gradient geometry. Our framework makes two key changes over prior art in private adaptive optimization: (1) Rather than privatizing the gradients first and then applying preconditioners, we show that transforming the gradients prior to privatization can reduce detrimental impacts of noise; (2) To perform gradient transformations, we explore using simple, easily obtainable side information in the data. We discuss two practical scenarios for obtaining such information below.

With Public Data. A natural choice for side information is to use a small amount of public data generated from a similar distribution as the private data, a common assumption in private optimization (Amid et al., 2021; Asi et al.,

[1]Carnegie Mellon University [2]Google DeepMind [3]Google Research. Correspondence to: Tian Li <tianli@cmu.edu>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

2021; Kairouz et al., 2021a; Zhou et al., 2021; 2020). In practice, public data could be obtained through proxy data or from 'opt-out' users who are willing to share their information (Kairouz et al., 2021a; Aldaghri et al., 2021). Indeed, the notion of heterogeneous DP where subsets of samples require zero or weak privacy has been extensively studied in prior works (e.g., Alaggan et al., 2015; Jorgensen et al., 2015). Another line of works is to assume that the gradients are low rank, and then use public data to estimate this gradient subspace—thereby mitigating some of the earlier discussed poor performance of DP in high-dimensional regimes. We do not consider using public data for this purpose in this work, as such a low-rank assumption might not hold in practice, particularly for the problems settings where adaptive optimizers are known to excel (Asi et al., 2021). Instead, we propose to use the public data more directly: we estimate gradient statistics on public data at each iteration, and then apply these statistics as a preconditioner *before* privatizing the gradients. Despite the simplicity of this procedure, we unaware of any work that has explored it previously.

Without Public Data. Of course, there may also be applications where it is difficult to obtain public data, particularly data that follows the same distribution as the private data. In such scenarios, our insight is that for many applications, in lieu of public data we may have access to some common knowledge about the training data that can be (i) computed before training, and (ii) used to improve optimization performance in both private and non-private settings. For instance, in many language tasks, certain aggregate statistics (e.g., frequency) of different words/tokens are common knowledge or may be easily computed prior to training, and serve as reasonably good estimates of the predictiveness of each feature. `AdaDPS` considers leveraging such simple heuristics to precondition the gradients. Perhaps surprisingly, in our experiments (Section 6), we demonstrate that the performance of `AdaDPS` when scaling gradients via these simple statistics (such as feature frequency) can even match the performance of `AdaDPS` with public data.

We summarize our main contributions below.

- We propose a simple yet effective framework, `AdaDPS`, to precondition the gradients with non-sensitive side information before privatizing them to realize the full benefits of adaptive methods in private training. Depending on the application at hand, we show that such side information can be estimated from either public data or some common knowledge about the data (e.g. feature frequencies or TF-IDF scores in NLP applications).
- We analyze `AdaDPS` and provide convergence guarantees for both convex and non-convex objectives. In convex cases, we analyze a specific form of `AdaDPS` using RMSProp updates to provably demonstrate the benefits of our approach relative to differentially private SGD when the gradients are sparse.
- Empirically, we evaluate our method on a set of real-world datasets. `AdaDPS` improves the absolute accuracy by 7.7% on average compared with strong baselines under the same privacy budget, and can even achieve similar accuracy as adaptive methods in non-private settings. We additionally demonstrate how to apply `AdaDPS` to the application of federated learning, where it outperforms existing baselines by a large margin.

2. Related Work

There are many DP algorithms for machine learning, including object perturbation, gradient perturbation, and model perturbation. In this work, we focus on the popular gradient perturbation method with Gaussian mechanisms (Dwork et al., 2014). Without additional assumptions on the problem structure, DP algorithms can suffer from $O(\frac{\sqrt{d}}{n\varepsilon})$ excess empirical risk where d is the dimension of the model parameters and n is the number of training samples (Bassily et al., 2014). While it is possible to mitigate such a dependence in the unconstrained setting (Kairouz et al., 2021a; Song et al., 2021) or assuming oracle access to a constant-rank gradient subspace (Zhou et al., 2021; Kairouz et al., 2021a), we do not focus on such a setting in this work, as it does not align with the settings in which adaptive methods have been designed (Duchi et al., 2011; Asi et al., 2021). Recent work (Amid et al., 2021) proposes to use public data differently (evaluating public loss as the mirror map in a mirror descent algorithm) to obtain dimension-independent bounds. However, this method do not account for gradient preconditioning, as their approximation is a linear combination of private and public gradients. We empirically verify `AdaDPS`'s superior performance in Section 6 (Table 2).

In the context of adaptive differentially private optimization, we note that 'adaptivity' may have various meanings. For example, Andrew et al. (2021) adaptively set the clipping threshold based on the private estimation of gradient norms, which is out of the scope of this work. Instead, our work is related to a line of work that aims to develop and analyze differentially private variants of common adaptive optimizers (e.g., private AdaGrad, private Adam), which mostly focus on estimating gradient statistics from noisy gradients (Zhou et al., 2020; Asi et al., 2021; Pichapati et al., 2019). They differ from `AdaDPS` in the order of preconditioning and privatization, the techniques used to approximate the gradient geometry, and the convergence analysis (see Section 5 for details). In empirical evaluation (Section 6), we compare `AdaDPS` with these works on diverse tasks and show that even `AdaDPS` without public data can outperform them.

3. Preliminaries and Setup

In terms of privacy formulations, we consider classic sample-level DP in centralized settings (Section 6.1), and a variant of it—*user-level DP*—in distributed/federated environments (Section 6.2). We define both more formally below.

Definition 1 (Differential privacy (Dwork et al., 2006))**.** *A randomized algorithm $\mathcal{M}$ is (ε, δ)-differentially private if for all neighbouring datasets D, D' differing by one element, and every possible subset of outputs O,*

$$\Pr(\mathcal{M}(D)\in O)\leq e^{\varepsilon}\Pr(\mathcal{M}(D')\in O)+\delta.$$

Within DP, neighbouring datasets can be defined in different ways depending on the application of interest. In this work, we also apply AdaDPS to federated learning (Section 6.2), where differential privacy is commonly defined at the granularity of users/devices (McMahan et al., 2018; Kairouz et al., 2021c), as stated below.

Definition 2 (User-level DP for federated learning (McMahan et al., 2018))**.** *A randomized algorithm $\mathcal{M}$ is (ε, δ)-differentially private if for all datasets U, U' differing by one user, and every possible subset of outputs O,*

$$\Pr(\mathcal{M}(U)\in O)\leq e^{\varepsilon}\Pr(\mathcal{M}(U')\in O)+\delta.$$

In centralized empirical risk minimization, our goal is to learn model parameters $w \in \mathbb{R}^d$ to fit n training samples $\{x^i\}_{i\in[n]}$: $\min_w F(w) = \frac{1}{n}\sum_{i=1}^n f(x^i; w)$, where $f(\cdot)$ is the individual loss function. Optionally, there may exist pubic data denoted as x_{pub}, which does not overlap with $\{x^i\}_{i\in[n]}$. We focus primarily on the classic centralized training, and later on extend our approach to federated settings (Objective (1)) (McMahan et al., 2017).

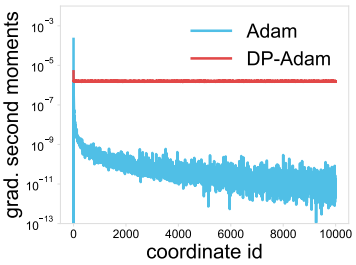

Figure 2. The estimates of gradient statistics (e.g., second moments) in private adaptive methods (e.g., DP-Adam) are noisy and may become uninformative of the relative importance of coordinates.

In gradient-based optimization, adaptive optimizers and their properties have been extensively studied (e.g., Mukkamala & Hein, 2017; Reddi et al., 2018a; Duchi et al., 2011; Kingma & Ba, 2015). They effectively result in coordinate-wise learning rates, which can be advantageous for many learning problems. The preconditioners can be estimated via a moving average of mini-batch gradients (as in, e.g., Adam) or simply by calculating the sum of gradients so far (AdaGrad). In private settings, estimating the required statistics on noisy gradients can introduce significant noise (Figure 2), making these methods less effective. To address this we introduce AdaDPS in the next section, and setup some notation below.

Notations. For vectors $u, v \in \mathbb{R}^d$, we use $u + v$ for coordindate-wise addition, and $\frac{u}{v}$ for coordinate-wise division. For a vector v and scalar a, $v + a$ denotes adding a to every dimension of v. For any vector v, v_j always denotes the j-th coordinate of v. For example, $g_j^{i,t}$ refers to the j-th coordinate of gradient $g^{i,t}$. We use $\|\cdot\|_M$ to denote the matrix norm defined as $\|\cdot\|_M := \sqrt{\langle\cdot, M\cdot\rangle}$ for a symmetric and positive definite matrix $M \in \mathbb{R}^{d\times d}$, or a diagonal matrix with non-negative diagonal entries $M \in \mathbb{R}^d$.

4. AdaDPS: Private Adaptive Optimization with Side Information

Gradient-based private optimization methods usually update the model parameters with noisy gradients at each iteration and then release the private final models (Abadi et al., 2016). To control the sensitivity of computing and summing individual gradients from a mini-batch, methods based on the subsampled Gaussian mechanism typically first clip each individual gradient and then add i.i.d. zero-mean Gaussian noise with variance determined by the clipping threshold and the privacy budget. To use adaptivity effectively, in AdaDPS, we instead propose first preconditioning the raw gradients with side information estimated either on public data or via some auxiliary knowledge, and then applying the Gaussian mechanism with noise multiplier σ on top. AdaDPS in centralized training is summarized in Algorithm 1. Note that AdaDPS is a general framework in that it incorporates a set of private adaptive methods. As described in Algorithm 1, the functions ϕ, φ, and $\mathcal{A}$ abstract a set of updating rules of different adaptive methods. Next, we describe the algorithm in more detail and instantiate ϕ, φ, and $\mathcal{A}$.

Option 1 (With Public Data). We first consider estimating gradient statistics based on public data. Functions ϕ, φ, and $\mathcal{A}$ can define a very broad a set of common adaptive updating rules, as shown below.

- *Adam*: A^t is the square root of the second moment estimation, and M^t is the first moment estimation; with $\mathcal{A}(g^{i,t}, A^t, M^t) = \frac{\beta^t g^{i,t} + (1-\beta^t)M^t}{A^t}$ for some moving averaging parameter β^t.
- *AdaGrad*: The update corresponds to $M^t = \mathbf{0}$, $(A^t)^2 = (A^{t-1})^2 + (\hat{g}^t)^2$, and $\mathcal{A}(g^{i,t}, A^t, M^t) = \frac{g^{i,t}}{A^t}$.

Algorithm 1: AdaDPS

Input: T, batch size b, noise multiplier σ, clipping threshold C, initial model $w^1 \in \mathbb{R}^d$, side information $A^t \in \mathbb{R}^d$, learning rate α^t, potential momentum buffer $M^0 \in \mathbb{R}^d$

for $t = 1, \cdots, T-1$ **do**

Uniformly sample a mini-batch B ($|B|$=b) from the training set and get b gradients:

$$g^{i,t} \leftarrow \nabla f(x^i; w^t),\ i \in B$$

Option 1: With public data x_{pub}

Uniformly sample a mini-batch B' ($|B'|$=b) from x_{pub}, get gradients, and update A^t and M^t with recurrence ϕ and φ respectively:

$$\hat{g}^t \leftarrow \frac{1}{b}\sum_{j \in B'} \nabla f(x^j; w^t),\ x^j \in x_{\text{pub}}$$

$$A^t \leftarrow \phi(A^{t-1}, \hat{g}^t), M^t \leftarrow \varphi(M^{t-1}, \hat{g}^t),$$

Option 2: Without public data

A^t estimated via heuristics

Precondition individual gradients by $\mathcal{A}$:

$$g^{i,t} \leftarrow \mathcal{A}(g^{i,t}, A^t, M^t)$$

Privatize preconditioned gradients:

$$\tilde{g}^t \leftarrow \frac{1}{b}\sum_{i \in B} \text{clip}\left(g^{i,t}, C\right) + \frac{1}{b}\mathcal{N}\left(0, \sigma^2 C^2\right),$$

where $\text{clip}(g, C)$ clips a vector g to L_2 norm C

Update the model parameter w as

$$w^{t+1} \leftarrow w^t - \alpha^t \tilde{g}^t$$

return w^T

- *RMSProp*: A^t is the square root of the second moment estimation with $M^t = \mathbf{0}$. And $\mathcal{A}(g^{i,t}, A^t, M^t) = \frac{g^{i,t}}{A^t}$.

We note that the AdaDPS framework can potentially incorporate other adaptive optimizers, beyond what is listed above. In our analysis and experiments, we mainly focus on using RMSProp updates to obtain the preconditioner, because AdaGrad which sums up gradients in all iterations so far in the denominator, often has poor practical performance, and Adam needs to maintain an additional mean estimator. However, we also evaluate the use of Adam within AdaDPS in Table 6 in Appendix C, showing that it yields similar improvements as RMSProp across all datasets. In Section 5, we analyze the convergence of A^t as $\mathbb{E}[(\hat{g}^t)^2]$, and prove that in sparse settings, AdaDPS allows the addition of less noise under the same privacy budget.

Option 2 (Without Public Data). When there is no public data available, we develop simple and effective heuristics to determine which coordinates are more predictive based on non-sensitive side information. In particular, for generalized linear models in NLP applications, we set A^t to be (i) proportional to the frequency of input tokens, or (ii) proportional to the TF-IDF values of input tokens. Follow a similar analysis as that of Option 1, we provide theoretical justification in Theorem 3 in Section 5.1.1. While these are simple approaches to remove the dependence on public data, we find that they can significantly outperform DP-SGD for real-world tasks with several million model parameters (Section 6.1).

Privacy guarantees. We now state the differential privacy guarantees of Algorithm 1. As the side information A^t (as well as the potential momentum buffer M^t) is non-sensitive, its privacy property directly follows from previous results (Abadi et al., 2016).

Theorem 1 (Privacy guarantee of Algorithm 1 (Abadi et al., 2016))**.** *Assume the side information A^t is non-sensitive. There exist constants c_1 and c_2 such that for any $\varepsilon < c_1 b^2 T/n^2$, Algorithm 1 is (ε, δ)-differentially private for any $\delta > 0$ if $\sigma \geq c_2 \frac{b\sqrt{T\log(1/\delta)}}{n\varepsilon}$.*

4.1. Intuition for A^t

In this section we provide further intuition for the AdaDPS framework. When A^t is an all-ones vector, AdaDPS reduces to the normal DP-SGD algorithm. Otherwise, A^t is indicative of how informative each coordinate is. Intuitively, suppose clipping does not happen and the public data come from the same distribution as private data so that for the RMSProp preconditioner, we have $A^t = \sqrt{\mathbb{E}[(g^{i,t})^2]}$. Then the effective transformation on each individual gradient $g^{i,t}$ is $\frac{g^{i,t} + \mathcal{N}\left(0, \sigma^2 C^2 \mathbb{E}[(g^{i,t})^2]\right)}{\sqrt{\mathbb{E}[(g^{i,t})^2]}}$. This can be viewed as first adding non-isotropic noise proportional to the second moment of gradients, and then applying RMSProp updates, which is beneficial as coordinates with higher second moments are more tolerant to noise. Therefore, AdaDPS could improve privacy/utility tradeoffs via adding coordinate-specific noise (formalized in Theorem 2).

We next consider a toy example to highlight one of the regimes where AdaDPS (or, adaptive methods) is particularly effective. Consider a linear regression task with the objective $\min_{w \in \mathbb{R}^{500}} \frac{1}{2n}\sum_{i \in [n]}(w^\top x^i - y^i)^2$ where $n = 1,000$ and each sample $x^i \in \mathbb{R}^{500}$. In many real-world applications, the tokens (features) are sparse and their frequencies follow heavy-tailed distributions. Without loss of generality, we assume the first 10% features are frequent and uninformative; and the later 90% rare and informative. Let the j-th feature of all data points be sampled from a random variable $x_j \in \{0, 1\}$. Features and the underlying true w are generated as follows:

$$\Pr(x_j{=}1){=}\begin{cases} 0.9, & j{\leq}50 \\ 0.01, & j{>}50 \end{cases},\quad w_j{=}\begin{cases} 0.01, & j{\leq}50 \\ 1.0, & j{>}50 \end{cases}.$$

Labels are generated by $y^i=\langle w,x^i\rangle+b^i$ where $b^i\sim\mathcal{N}(0,0.01)$. For AdaDPS, we assume model engineers know which words are more frequent, thus setting $A_j^t=1$ for $j\leq 50$ and $A_j^t=0.01$ otherwise for all t. Using larger learning rates on informative coordinates, side information helps to improve optimization performance dramatically (see results in Figure 3).

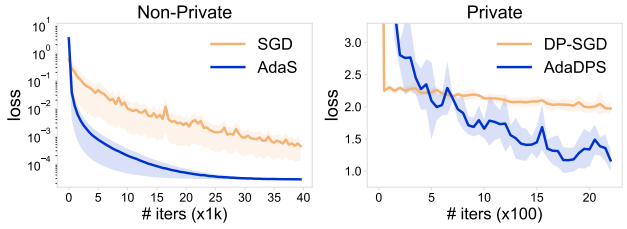

Figure 3. Training loss on the linear regression problem described in Section 4 (averaged over five runs). We tune optimal learning rates separately for each method. Private training (right) achieves $(4.13,10^{-3})$-DP.

Comparison to Asi et al. (2021). The most related work to ours is Asi et al. (2021), which adds non-isotropic noise which lies in a non-uniform ellipsoid to the gradients, and (optionally) transforms the resulting gradients with the demoninator used in AdaGrad. AdaDPS differs from their approach in several ways, as we (i) first precondition then add noise (as opposed to the other way round), (ii) consider a broader class of preconditioners (beyond AdaGrad), and (iii) make the approaches to estimating gradient geometry in Asi et al. (2021) more explicit in lieu of public data (as discussed in previous sections). Empirically, we compare with another state-of-the-art method (Amid et al., 2021) which outperforms Asi et al. (2021), and demonstrate AdaDPS's superior performance (Section 6, Table 2).[1]

5. Convergence Analysis

We now analyze the convergence of AdaDPS (Algorithm 1) with and without public data, in both convex (Section 5.1) and non-convex (Section 5.2) settings. When there is public data available, we prove that AdaDPS adds less noise (plus, with noise proportional to the magnitude of gradients) compared with DP-SGD. Our theory extends previous proofs in related contexts, but considers stochastic gradients, adding random Gaussian noise to the processed gradients, and estimating the preconditioner on public data (as opposed to updating it with the raw gradients on training data). When there is no public data, we present convergence results for general A^t, covering the heuristics used in practice.

[1]We do not compare with Asi et al. (2021) directly as the code is not publicly available.

5.1. Convex Convergence

For convex functions, we define the optimal model w^* as $w^*\in\operatorname{argmin}_w F(w)$. First we state a set of assumptions that are used in the analyses.

Assumption 1. *There exists a constant D such that $\|w^t-w^*\|_2\leq D$ for any $t\in[T]$.*

Assumption 2. *There exists a constant C such that $\left\|\frac{g^{i,t}}{A^t}\right\|_2\leq C$ for any $t\in[T]$ and $i\in[n]$.*

Assumption 3. *Denote $g^t:=\frac{1}{b}\sum_{i\in B}g^{i,t}$. There exists a constant $a\in(0,1]$ such that for any $j\in[d]$ and $t\in[T]$, $a(\hat{g}_j^t)^2\leq(g_j^t)^2\leq\frac{1}{a}(\hat{g}_j^t)^2$ holds.*

Assumption 2 aims to bound the L_2 norm of the transformed gradient, thus resulting in bounded L_2 sensitivity on the operation of calculating and averaging (scaled) individual gradients from a mini-batch. Assuming bounded stochastic gradient norm is standard in prior works on convex and non-convex private optimization (e.g., Kairouz et al., 2021a; Zhou et al., 2020). Assumption 3 bounds the dissimilarity between public and private data.

Within the framework of AdaDPS, we explore the convergence of the RMSProp preconditioner where $A^t=\sqrt{\mathbb{E}[(\hat{g}^t)^2]}+\epsilon^t$ (with public data) where ϵ^t is some small constant or A^t is obtained via side information heuristics. We first look at the case where there exist public data, and therefore the exact updating rule at the t-th iteration is:

$$\hat{g}^t,g^t\leftarrow\frac{1}{b}\sum_{j\in B'}\nabla f(x^j;w^t)\ (x^j\in x_{\text{pub}}),\frac{1}{b}\sum_{i\in B}\nabla f(x^i;w^t),$$

$$v^t\leftarrow\beta^t v^{t-1}+(1-\beta^t)(\hat{g}^t)^2,\ A^t\leftarrow\sqrt{v^t}+\epsilon^t,$$

$$w^{t+1}\leftarrow w^t-\alpha^t\left(\frac{g^t}{A^t}+N\right),\ N\sim\frac{1}{b}\mathcal{N}(0,\sigma^2C^2).$$

Theorem 2 below states the convergence guarantees.

Theorem 2. *Assume $F(w)$ is a convex function w.r.t. w. Let Assumptions 1-3 hold. Additionally, choose β^t such that $1-\frac{\gamma}{t}\geq\beta^t\geq 1-\frac{1}{t}$ holds for some $\gamma\in(0,1]$; and let $\sqrt{t+1}\epsilon^{t+1}\geq\sqrt{t}\epsilon^t$ for any t. After running Algorithm 1 using learning rates $\alpha^t=\frac{\alpha}{\sqrt{t}}$ with public data for T iterations, we have*

$$\min_{t\in[T]}\mathbb{E}[F(w^t)]-F^*\leq\frac{G}{\sqrt{T}}\sum_{j=1}^{d}\mathbb{E}\left[A_j^T\right]+\frac{\alpha}{\sqrt{T}}\max_{t\in[T]}\mathbb{E}\left[\|N\|_{A^t}^2\right],$$

where $F^*:=F(w^*)$, $G=\frac{D^2}{2\alpha}+\frac{\alpha(2-\gamma)}{a\gamma}$, and $A_j^T=\sqrt{v_j^T}+\epsilon^T$.

The full proof is deferred to Appendix A.1. Our proof extend the proof of the original RMSProp method with full gradients in Mukkamala & Hein (2017) to stochastic and

private settings. From Theorem 4, we see that the first term in the bound is standard for the RMSProp optimizer, and the last term is due to noise added to ensure differential privacy. Fixing the noise multiplier σ, the second term depends on the clipping value C and the preconditioner A^t. We see that when the gradients are sparse, it is likely that the added DP noise would be significantly reduced. In other words, to guarantee overall (ε,δ)-differential privacy by running T total iterations, we can set $\sigma^2=O\left(\frac{b^2T\log(1/\delta)}{n^2\varepsilon^2}\right)$ and $T=O\left(\frac{n^2\varepsilon^2}{\max_{t\in[T]}\mathbb{E}\left[\|A^t\|_1\right]\log(1/\delta)}\right)$. The convergence rate therefore becomes

$$\min_{t\in[T]}\mathbb{E}[F(w^t)]-F^*\leq O\left(\frac{\sqrt{\max_{t\in[T]}\mathbb{E}[\|A^t\|_1]}}{n\varepsilon}\right).$$

When gradients are sparse (hence $\max_{t\in[T]}\mathbb{E}[\|A^t\|_1]<d$), the amount of noise added will be significantly smaller compared with that of vanilla DP-SGD to guarantee the same level of privacy. This highlights one regime where `AdaDPS` is particularly useful, though `AdaDPS` also yield improvements in other settings with dense gradients (Table 6 in the appendix). Here, Theorem 2 assumes access to $\mathbb{E}[(\hat{g}^t)^2]$. When there is no public data available, we leverage other easily obtainable side information to determine fixed A^t prior to training, as analyzed in the next section.

5.1.1. Fixed A^t

Theorem 3. *Let assumptions in Theorem 2 hold. Running Algorithm 1 using learning rates $\alpha^t=\frac{\alpha}{\sqrt{t}}$ without public data with side information $A\in\mathbb{R}^d$ for T iterations gives*

$$\min_{t\in[T]}\mathbb{E}[F(w^t)]-F^*\leq O\left(\frac{\alpha R+1}{\sqrt{T}}\sum_{j=1}^{d}A_j+\frac{\alpha}{\sqrt{T}}\mathbb{E}\left[\|N\|_A^2\right]\right),$$

where $R:=\max_{j,t}\frac{\mathbb{E}[(g_j^t)^2]}{A_j^2}$ and $g^t:=\frac{1}{b}\sum_{i\in B}g^{i,t}$.

From Theorem 3 above, we observe that A should be chosen such that both R and $\sum_{i=j}^{d}A_j$ are minimized. For a large class of generalized linear models, we are able to obtain appropriate A based on the feature space information to control the values of R, as discussed in the following.

Considering generalized linear models. Under the model parameter $w\in\mathbb{R}^d$, for any x^i, the gradients are $c(x^i;w)x^i$ where $c(x^i;w)\in\mathbb{R}$ is a function of x^i and w. We assume that for any $i\in[n]$ and w, there exists a constant $c_{\max}$ such that $|c(x^i;w)|\leq c_{\max}$. One natural choice of A is to set $A_j=\sqrt{\mathbb{E}[x_j^2]}+\epsilon$ for each coordinate $j\in[d]$ (such that $R\leq c_{\max}^2$), which could improve the noise term $\mathbb{E}\left[\|N\|_A^2\right]$ when the features are sparse. Nevertheless, $\mathbb{E}[x^2]$ can be unrealistic to obtain prior to training. Instead, one side information of choice is $\mathbb{E}[x]$, which implies how rare the raw features are in some NLP applications. Let $A_j=\mathbb{E}[|x_j|]+\epsilon$ $(j\in[d])$. Then

$$R=\max_{j,t}\frac{\mathbb{E}\left[(g_j^t)^2\right]}{\left(\mathbb{E}[|x_j|]+\epsilon\right)^2}\leq\max_{j,t}\frac{c_{\max}^2\mathbb{E}[x_j^2]}{(\mathbb{E}[|x_j|])^2+\epsilon^2}.$$

To reason about how large R is, we consider a simple setup where each feature takes the value of $v>0$ with probability p, and 0 with probability $1-p$. It is straightforward to see the scaling of the last two terms in the convergence bound in Theorem 3:

$$R\sum_{j=1}^{d}A_j=O\left(\frac{1}{p}\right)O(dpv)=O(dv),$$

$$\mathbb{E}\left[\|N\|_A^2\right]=\sigma^2C^2\sum_{j=1}^{d}A_j=O\left(\max_{i,t}\|g^{i,t}\|^2\frac{dpv}{(pv+\epsilon)^2}\right).$$

$\mathbb{E}\left[\|N\|_A^2\right]$ will reduce to $O(d)$ if the gradients are sparse in a certain way, i.e., $\max_{i,t}\|g^{i,t}\|^2=O(p)$. Note that only using this simple first moment information ($A_j=\mathbb{E}[|x_j|]+\epsilon$), we are not able to obtain a constant improvement in the convergence bound as in Theorem 2 with public data. However, we empirically demonstrate that these ideas can be effective in practice in Section 6.

5.2. Non-Convex Convergence

In addition to convex problems, we also consider convergence to a stationary point for non-convex and smooth functions. We introduce additional assumptions in the following.

Assumption 4. *$F(\cdot)$ is L-smooth.*

Assumption 5. *The expectation of stochastic gradient variance is bounded, i.e., $\mathbb{E}[\|g_j^{i,t}-\mathbb{E}[g_j^{i,t}]\|_2^2]\leq\tau_j^2$ for all i,t,j. Denote $\tau^2:=(\tau_1^2,\cdots,\tau_d^2)\in\mathbb{R}^d$.*

Assumption 4 together with Assumption 1 implies that there exists a constant that bounds $\|\nabla F(w^t)\|$ for any t, which we denote as B.

Theorem 4. *Let Assumptions 1-5 hold. After running Algorithm 1 with public data for T iterations using a constant learning rate α and a constant ϵ, choosing the constants to satisfy $\alpha\leq\frac{\epsilon}{2L}$ and $B\sqrt{1-\beta}\leq\frac{\sqrt{a}\epsilon}{4}$, we have*

$$\frac{1}{T}\sum_{t\in[T]}\mathbb{E}\left[\|\nabla F(w^t)\|^2\right]\leq O\left(\frac{1}{T}\right)+O\left(\frac{\|\tau^2\|_1}{b}+\frac{\sigma^2}{b^2}\right).$$

The proof is mostly extended from Adam's proof in Reddi et al. (2018a) (see Appendix A.2 for complete steps). When the batch size increases, both the stochastic gradient noise and differential privacy noise would be reduced.

Theorem 5. *Let Assumptions 1-5 hold. After running Algorithm 1 with a fixed A as prior information for T iterations using a constant learning rate α and a constant ϵ, choosing the constants to satisfy $\alpha \leq \frac{\epsilon}{L}$, we have*

$$\frac{1}{T}\sum_{t\in[T]}\mathbb{E}\left[\|\nabla F(w^t)\|^2_{A^{-1}}\right] \leq O\left(\frac{1}{T}\right) + O\left(\frac{\tau^2}{A}\frac{1}{b} + \frac{\sigma^2}{b^2}\right).$$

6. Experiments

We evaluate the performance of AdaDPS in both centralized (Section 6.1) and federated (Section 6.2) settings for various tasks and models. In centralized training, we investigate two practical scenarios for obtaining side information with and without public data (Section 6.1.1 and 6.1.2). We describe our experimental setup below; details of datasets, models, and hyperparameter tuning are described in Appendix B. Our code is publicly available at github.com/litian96/AdaDPS.

Datasets. We consider common benchmarks for adaptive optimization in centralized or federated settings (Amid et al., 2021; Reddi et al., 2018a; 2021) involving varying types of models (both convex and non-convex) and data (both text and image data). Both linear and non-convex models contain millions of learnable parameters.

Hyperparameters. We fix the noise multiplier σ for each task, and select an individual (fixed) clipping threshold for each differentially private run. To track the privacy loss (to ensure (ε,δ)-DP), we add the same amount of noise to all compared methods, set the δ value to be the inverse of the number of all training samples, and compute ε using Rényi differential privacy (RDP) accountant for the subsampled Gaussian mechanism (Mironov et al., 2019).

6.1. Centralized Training

Common Baselines. One can directly privatize an adaptive optimizer by *first privatizing* the raw gradients, and *then applying that adaptive method* on top of noisy gradients (Zhou et al., 2020). We consider these baselines named DP-Adam or DP-RMSProp where the adaptive optimizer is chosen to be Adam or RMSProp (same as DP-Adam appearing in previous sections). As the empirical results of DP-Adam and DP-RMSProp are very similar (Table 6 in the appendix), in the main text, we mainly compare AdaDPS with DP-Adam (Zhou et al., 2020) and DP-SGD (Abadi et al., 2016). For completeness, we present the exact DP-Adam algorithm in Appendix B.

6.1.1. With public data

In the main text, we set the public data size to be 1% of training data size. We further explore the effects of public data size empirically in Table 10, Appendix C. Next, we present results of comparing AdaDPS with several baselines, and results using both in-distribution (ID) and out-of-distribution (OOD) data as public data to estimate the preconditioner.

Preconditioning *Noisy* Gradients with Public Data. In addition to DP-SGD and DP-Adam mentioned in Section 6.1, we consider another method of preconditioning the *noisy* gradients with second moment estimates obtained from *public data*. Specifically, the updating rule at iteration t is

$$\tilde{g}^t \leftarrow \frac{1}{b}\sum_{i\in B}\text{clip}\left(g^{i,t}, C\right) + \frac{1}{b}\mathcal{N}(0, \sigma^2 C^2),$$

$$\tilde{g}^t \leftarrow \frac{\tilde{g}^t}{\sqrt{\mathbb{E}[(\hat{g}^t)^2] + \epsilon^t}}, \text{ where } \hat{g}^t := \nabla f(x; w^t) \text{ for } x \in x_{\text{pub}}.$$

We call this adaptive baseline DP-R-Pub, which is equivalent to standard DP-RMSProp but using public data to estimate the preconditioner. Comparing with this method directly reflects the importance of the order of preconditioning and privatization in AdaDPS.

Results with in-distribution proxy data (randomly sampled from training sets) are shown in Figure 4 and Table 1 below. We see that across three datasets, (i) DP-Adam does not necessarily outperform DP-SGD all the same, (ii) AdaDPS improves over all baselines significantly, including DP-R-Pub. Full results involving DP-RMSProp and AdaDPS with Adam as the updating rule are presented in Table 6.

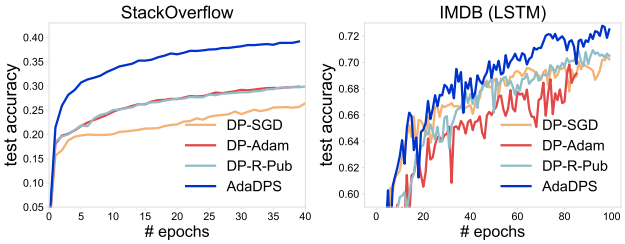

Figure 4. Test accuracies of baselines and AdaDPS assuming access to public data. ε values on these two datasets are 0.84 and 2.8, respectively. AdaDPS significantly improves test performance, even reaching an accuracy much higher than the accuracy of SGD in non-private training on StackOverflow.

Methods	**Loss** $\times 100$ (σ=1)	**Loss** $\times 100$ (σ=0.75)
DP-SGD	5.013 (.001)	3.792 (.001)
DP-Adam	3.697 (.020)	3.286 (.016)
AdaDPS	**3.566** (.008)	**3.158** (.003)

Table 1. Test reconstruction loss (mean and standard deviation across three runs) on MNIST under a deep autoencoder model. σ=1 and σ=0.75 correspond to ε=1.6 and ε=3, respectively. DP-Adam works well in this task compared with DP-SGD. AdaDPS improves over DP-Adam.

For completeness, we also evaluate AdaDPS on the MNIST image classification benchmark, and observe that it yields $0.5\%-2\%$ accuracy improvements (Table 6), depending on which specific adaptive method to use.

Comparisons with Amid et al. (2021). We compare AdaDPS with one recent work, PDA-DPMD, which is the state-of-the-art that leverages public data to improve privacy/utility tradeoffs in a mirror descent framework (Amid et al., 2021). We take their proposed approximation, where the actual gradients are a convex combination of gradients on private and public data. As this approximation does not precondition gradients, PDA-DPMD could underperform AdaDPS in the tasks where adaptivity is critical, as shown in Table 2 below.

Datasets	Metrics	PDA-DPMD w/ public	AdaDPS w/ public	AdaDPS w/o public
IMDB	accuracy	0.62	**0.80**	0.75
StackOverflow	accuracy	0.33	**0.40**	**0.41**
MNIST	loss	0.039	**0.036**	—

Table 2. Comparison with a recent method (PDA-DPMD) using public data in private mirror descent. AdaDPS outperforms PDA-DPMD due to preconditioning.

Out-Of-Distribution Public Data As mentioned in Section 1, public data could be obtained via a small amount of proxy data or 'opt-out' users that do not need privacy protection. We consider two practical cases where we use OOD data to extract side information. For IMDB sentiment analysis, we use a small subset of Amazon reviews[2] (Zhang et al., 2015) as public data (1% the size of IMDB). Amazon reviews study a more fine-grained 5-class classification problem on product reviews, and we map labels $\{1, 2\}$ to 0 (negative), and labels $\{4, 5\}$ to 1 (positive). For StackOverflow tag prediction task which consists of 400 users with different styles and interested topics, we simply hold out the first four of them to provide public data. We show results in Table 3 below. We see that the preconditioners obtained from out-of-distribution but related public data are fairly effective.

Datasets	DP-SGD	AdaDPS OOD public	AdaDPS ID public
IMDB	0.63	**0.79**	**0.80**
StackOverflow	0.28	**0.40**	**0.40**

Table 3. Using small out-of-distribution data as public data achieves the same improvements. For IMDB, we leverage Amazon reviews data (1% the size of IMDB) as public data. For StackOverflow, we hold out 1% users as those who opt out of privacy training.

[2] figshare.com/articles/dataset/Amazon_Reviews_Full/13232537/1

6.1.2. Without public data

When it is difficult to obtain public data that follow sufficiently similar distribution as private training data, we explore two specific heuristics as side information tailored to language tasks: token frequencies and TF-IDF values of input tokens (or features). These statistics are always known as open knowledge, thus can be used as an approximate how important each feature is. We compare AdaDPS with DP-SGD and DP-Adam described before.

A^t Based on Token Frequencies. One easily obtainable side information is token frequencies, and we can set A^t $(t\in[T])$ to be proportional to that accordingly. Note that our implicit assumption here is that rare features are more informative than frequent ones. We investigate the logistic regression model on two datasets in Figure 4 below. Despite the simplicity, this simple method works well on StackOverflow and IMDB under a tight privacy budget, outperforming DP-SGD and DP-Adam significantly. Especially for StackOverflow, the test accuracy is the same as that of AdaDPSwith a small set of public data (Figure 4).

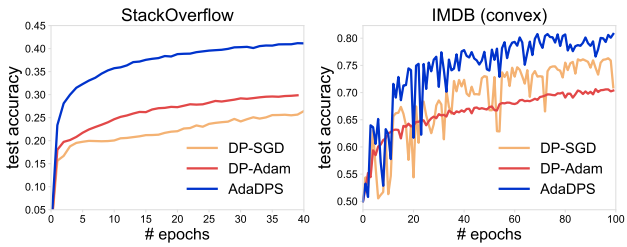

Figure 5. AdaDPS uses token frequencies as the side information, demonstrating superior performance than the baselines. Methods in each subfigure reach $(0.84, 4.2\times10^{-6})$- and $(2.8, 4\times10^{-5})$-DP.

A^t Based on TF-IDF Values. Another common criterion to measure the relevance of tokens to each data point in a collection of training data is TF-IDF values. With the presence of such information available in a data processing pipeline, we explore the effects of A^t being inversely proportional to TF-IDF scores for linear models. The results are reported in Table 4. The 'ideal' method refers to AdaDPS with RMSProp updates and the second moment estimated on clean gradients from private training data, which serves a performance upper bound. As expected, across all methods, TF-IDF features result in higher accuracies than BoW features. AdaDPS outperforms the baselines by $\sim$10% absolute accuracy, only slightly underperforming the 'ideal' oracle.

Remark (Side Information in Non-Private Training). The idea of using side information (with or without public data) can also improve the performance of vanilla SGD in non-private training, yielding similar accuracies as that of

Features	Methods			
	DP-SGD	DP-Adam	AdaDPS	*ideal*
BoW	0.62 (.02)	0.68 (.01)	**0.75** (.01)	*0.82 (.01)*
TF-IDF	0.68 (.01)	0.65 (.01)	**0.80** (.00)	*0.83 (.00)*

Table 4. We preprocess IMDB into two versions with either BoW or TF-IDF features, and report average test accuracy along with standard deviation across three runs. AdaDPS with A^t being inversely proportional to features' TF-IDF values outperforms the baselines of DP-SGD and DP-Adam by a large margin. AdaDPS also performs relatively closely to the 'ideal' upper bound.

adaptive methods. We report additional results along this line in Table 9 in the appendix.

6.2. Federated Learning

In this section, we discuss AdaDPS adapted to federated learning (FL) (learning statistical models over heterogeneous networks while keeping all data local) to satisfy user-level, global differential privacy (assuming a trusted central server). The canonical objective for FL is to fit a single model $w \in \mathbb{R}^d$ to data across a network of n devices:

$$\min_{w \in \mathbb{R}^d} F(w) = \sum_{i=1}^{n} p_i f_i(w), \tag{1}$$

where $f_i(w)$ is the empirical local loss on each device i, and p_i is some pre-defined weight for device i such that $\sum_{i=1}^{n} p_i = 1$, which can be $\frac{1}{n}$ or proportional to the number of local samples. In this work, we simply set $p_i = \frac{1}{n}$, $i \in [n]$. For this privacy-sensitive application, we assume there is no public data available.

Due to the practical constraints of federated settings (e.g., unreliable network conditions, device heterogeneity, etc), federated optimization algorithms typically randomly samples a small subset of devices at each communication round, and allows each device to run optimization methods locally (e.g., local mini-batch SGD) before sending the updates back to the server (McMahan et al., 2017). Adapting AdaDPS to federated learning is not straightforward, raising questions of applying preconditioning at the server side, the device side, or both (Wang et al., 2021). We empirically find that on the considered dataset, preconditioning the mini-batch gradients *locally at each iteration* demonstrates superior performance than preconditioning the entire model updates at the server side. The exact algorithm is summarized in Algorithm 3 in the appendix.

We investigate the same StackOverflow dataset described in Section 6.1, but follow its original, natural partition (by Stack Overflow users) for federated settings, one device per user. There are 400 devices in total for the subsampled version we use. We select 20 devices at each communication round and use a noise multiplier $\sigma = 0.3$. While we arrive at a large ε value for user-level DP, the final model could still be useful for defending against membership inference attacks in practice (Kairouz et al., 2021b).

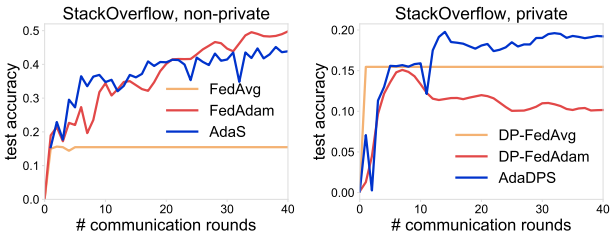

Figure 6. Test accuracy of StackOverflow in non-private and private settings under $(34, 0.0025)$ user-level DP (Definition 2). AdaDPS extended to federated learning (Algorithm 3 in the appendix) improves over baselines of DP-FedAvg (McMahan et al., 2018) and DP-FedAdam by 5% in terms of test accuracy.

In Figure 6, we plot test accuracy versus the number of communication rounds. AdaDPS has $\sim$5% higher test accuracy than other two methods. We note that in federated learning applications involving massive and unreliable networks, it is not always realistic to allow for uniform device sampling. Incorporating recent advances in DP without sampling (e.g., Kairouz et al., 2021b) to address this is left for future work.

7. Conclusion

In this work, we explored a simple and effective framework, AdaDPS, to realize the benefits of adaptive optimizers in differentially private learning via side information. Such information is used to precondition gradients before privatizing them. We analyzed the benefits of AdaDPS in terms of reducing the effective noise to reach similar privacy bounds, and empirically validated its superior performance across various tasks in both centralized and federated settings.

Acknowledgements

We thank Brendan McMahan and Abhradeep Thakurta for valuable discussions and feedback. The work of TL and VS was supported in part by the National Science Foundation Grant IIS1838017, a Google Faculty Award, a Facebook Faculty Award, the Private AI Collaborative Research Institute, and the CONIX Research Center. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the National Science Foundation or any other funding agency.

References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. Deep learning with differential privacy. In *Conference on Computer and*

Communications Security, 2016.

Alaggan, M., Gambs, S., and Kermarrec, A.-M. Heterogeneous differential privacy. *arXiv preprint arXiv:1504.06998*, 2015.

Aldaghri, N., Mahdavifar, H., and Beirami, A. Feo2: Federated learning with opt-out differential privacy. *arXiv preprint arXiv:2110.15252*, 2021.

Amid, E., Ganesh, A., Mathews, R., Ramaswamy, S., Song, S., Steinke, T., Suriyakumar, V. M., Thakkar, O., and Thakurta, A. Public data-assisted mirror descent for private model training. *arXiv preprint arXiv:2112.00193*, 2021.

Andrew, G., Thakkar, O., McMahan, H. B., and Ramaswamy, S. Differentially private learning with adaptive clipping. In *Advances in Neural Information Processing Systems*, 2021.

Asi, H., Duchi, J., Fallah, A., Javidbakht, O., and Talwar, K. Private adaptive gradient methods for convex optimization. In *International Conference on Machine Learning*, 2021.

Authors, T. T. F. TensorFlow Federated Stack Overflow dataset, 2019. URL https://www.tensorflow.org/federated/api_docs/python/tff/simulation/datasets/stackoverflow/load_data.

Bassily, R., Smith, A., and Thakurta, A. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *Symposium on Foundations of Computer Science*, 2014.

Chaudhuri, K., Monteleoni, C., and Sarwate, A. D. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 2011.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 2011.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference*, 2006.

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

Hinton, G., Srivastava, N., and Swersky, K. Rmsprop: Divide the gradient by a running average of its recent magnitude. *Neural Networks for Machine Learning, Coursera Lecture 6e*, 2012.

Jorgensen, Z., Yu, T., and Cormode, G. Conservative or liberal? personalized differential privacy. In *International Conference on Data Engineering*, 2015.

Kairouz, P., Diaz, M. R., Rush, K., and Thakurta, A. (nearly) dimension independent private erm with adagrad rates via publicly estimated subspaces. In *Conference on Learning Theory*, 2021a.

Kairouz, P., McMahan, B., Song, S., Thakkar, O., Thakurta, A., and Xu, Z. Practical and private (deep) learning without sampling or shuffling. In *International Conference on Machine Learning*, 2021b.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al. Advances and open problems in federated learning. *Foundations and Trends in Machine Learning*, 2021c.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

Maas, A., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. Learning word vectors for sentiment analysis. In *Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 2011.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *International Conference on Artificial Intelligence and Statistics*, 2017.

McMahan, H. B., Ramage, D., Talwar, K., and Zhang, L. Learning differentially private recurrent language models. *International Conference on Learning Representations*, 2018.

Mironov, I., Talwar, K., and Zhang, L. Rényi differential privacy of the sampled gaussian mechanism. *arXiv preprint arXiv:1908.10530*, 2019.

Mukkamala, M. C. and Hein, M. Variants of rmsprop and adagrad with logarithmic regret bounds. In *International Conference on Machine Learning*, 2017.

Pichapati, V., Suresh, A. T., Yu, F. X., Reddi, S. J., and Kumar, S. Adaclip: Adaptive clipping for private sgd. *arXiv preprint arXiv:1908.07643*, 2019.

Reddi, S., Zaheer, M., Sachan, D., Kale, S., and Kumar, S. Adaptive methods for nonconvex optimization. In *Advances in Neural Information Processing Systems*, 2018a.

Reddi, S., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečnỳ, J., Kumar, S., and McMahan, H. B. Adaptive federated optimization. *International Conference on Learning Representations*, 2021.

Reddi, S. J., Kale, S., and Kumar, S. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018b.

Song, S., Steinke, T., Thakkar, O., and Thakurta, A. Evading the curse of dimensionality in unconstrained private glms via private gradient descent. In *International Conference on Artificial Intelligence and Statistics*, 2021.

Wang, J., Charles, Z., Xu, Z., Joshi, G., McMahan, H. B., et al. A field guide to federated optimization. *arXiv preprint arXiv:2107.06917*, 2021.

Zhang, J., Karimireddy, S. P., Veit, A., Kim, S., Reddi, S. J., Kumar, S., and Sra, S. Why are adaptive methods good for attention models? In *Advances in Neural Information Processing Systems*, 2020.

Zhang, X., Zhao, J., and LeCun, Y. Character-level convolutional networks for text classification. *Advances in Neural Information Processing Systems*, 2015.

Zhou, Y., Chen, X., Hong, M., Wu, Z. S., and Banerjee, A. Private stochastic non-convex optimization: Adaptive algorithms and tighter generalization bounds. *arXiv preprint arXiv:2006.13501*, 2020.

Zhou, Y., Wu, Z. S., and Banerjee, A. Bypassing the ambient dimension: Private sgd with gradient subspace identification. In *International Conference on Learning Representations*, 2021.

A. Convergence

A.1. Proof for Theorem 2 (Convex cases)

Based on our assumptions, the updating rule becomes

$$g^t \leftarrow \frac{1}{b}\sum_{i\in B}\nabla f(x^i;w^t) \tag{2}$$

$$\hat{g}^t \leftarrow \frac{1}{b}\sum_{j\in B'}\nabla f(x^j;w^t),\ x^j \in x_{\text{pub}} \tag{3}$$

$$v^t \leftarrow \beta^t v^{t-1}+(1-\beta^t)(\hat{g}^t)^2 \tag{4}$$

$$A^t \leftarrow \sqrt{v^t}+\epsilon^t \tag{5}$$

$$w^{t+1} \leftarrow w^t-\alpha^t\left(\frac{g^t}{A^t}+N\right),\ N\sim\frac{1}{b}\mathcal{N}(0,\sigma^2C^2) \tag{6}$$

We extend the proof in Mukkamala & Hein (2017) to stochastic, private cases with preconditioner estimated on public data. Based on the updating rule, we have

$$\left\|w^{t+1}-w^*\right\|_{A^t}^2 \tag{7}$$

$$=\left\|w^t-\alpha^t(A^t)^{-1}g^t-\alpha^tN-w^*\right\|_{A^t}^2 \tag{8}$$

$$=\left\|w^t-w^*\right\|_{A^t}^2+\left\|\alpha^t(A^t)^{-1}g^t+\alpha^tN\right\|_{A^t}^2-2\left\langle w^t-w^*,\alpha^tg^t+\alpha^tA^tN\right\rangle \tag{9}$$

$$=\left\|w^t-w^*\right\|_{A^t}^2-2\alpha^t\left\langle g^t,w^t-w^*\right\rangle+(\alpha^t)^2\left\langle g^t,(A^t)^{-1}g^t\right\rangle-2\alpha^t\langle w^t-w^*,A^tN\rangle+(\alpha^t)^2\|N\|_{A^t}^2+2(\alpha^t)^2\langle g^t,N\rangle. \tag{10}$$

Rearranging terms gives

$$\left\langle g^t,w^t-w^*\right\rangle=\frac{\|w^t-w^*\|_{A^t}^2-\|w^{t+1}-w^*\|_{A^t}^2}{2\alpha^t}+\frac{\alpha^t}{2}\left\langle g^t,(A^t)^{-1}g^t\right\rangle-\langle w^t-w^*,A^tN\rangle+\frac{\alpha^t}{2}\|N\|_{A^t}^2+\alpha^t\langle g^t,N\rangle. \tag{11}$$

Take expectation on both sides conditioned on w^t,

$$\left\langle\nabla F(w^t),w^t-w^*\right\rangle=\frac{\mathbb{E}_t\left[\|w^t-w^*\|_{A^t}^2\right]-\mathbb{E}_t[\|w^{t+1}-w^*\|_{A^t}^2]}{2\alpha^t}+\frac{\alpha^t}{2}\mathbb{E}_t\left[\left\langle g^t,(A^t)^{-1}g^t\right\rangle\right]+\frac{\alpha^t}{2}\mathbb{E}_t\left[\|N\|_{A^t}^2\right], \tag{12}$$

where we have used the fact that N is a zero-mean Gaussian variable independent of g^t,w^t. Taking expectation on both sides and using the convexity of $F(\cdot)$:

$$\mathbb{E}[F(w^t)]-F(w^*)\le\frac{\mathbb{E}[\|w^t-w^*\|_{A^t}^2]-\mathbb{E}[\|w^{t+1}-w^*\|_{A^t}^2]}{2\alpha^t}+\frac{\alpha^t}{2}\mathbb{E}[\left\langle g^t,(A^t)^{-1}g^t\right\rangle]+\frac{\alpha^t}{2}\mathbb{E}\left[\|N\|_{A^t}^2\right]. \tag{13}$$

Applying telescope sum, we have

$$\sum_{t=1}^{T}\left(\mathbb{E}[F(w^t)]-F(w^*)\right) \tag{14}$$

$$\le\frac{\|w^1-w^*\|_{A^1}^2}{2\alpha_1}+\sum_{t=2}^{T}\left(\frac{\mathbb{E}\left[\|w^t-w^*\|_{A^t}^2\right]}{2\alpha^t}-\frac{\mathbb{E}\left[\|w^t-w^*\|_{A^{t-1}}^2\right]}{2\alpha_{t-1}}\right)+\sum_{t=1}^{T}\frac{\alpha^t}{2}\mathbb{E}\left[\left\langle g^t,(A^t)^{-1}g^t\right\rangle\right]+\sum_{t=1}^{T}\frac{\alpha^t}{2}\mathbb{E}\left[\|N\|_{A^t}^2\right]. \tag{15}$$

Let $\alpha^t=\frac{\alpha}{\sqrt{t}}$,

$$\sum_{t=1}^{T}\left(\mathbb{E}[F(w^t)]-F(w^*)\right) \tag{16}$$

$$\le\frac{\|w^1-w^*\|_{A^1}^2}{2\alpha}+\underbrace{\sum_{t=2}^{T}\frac{\mathbb{E}\left[\|w^t-w^*\|_{\sqrt{t}A^t-\sqrt{t-1}A^{t-1}}^2\right]}{2\alpha}}_{T_1}+\underbrace{\sum_{t=1}^{T}\frac{\alpha}{2\sqrt{t}}\mathbb{E}\left[\left\langle g^t,(A^t)^{-1}g^t\right\rangle\right]}_{T_2}+\sum_{t=1}^{T}\frac{\alpha}{2\sqrt{t}}\mathbb{E}\left[\|N\|_{A^t}^2\right]. \tag{17}$$

Let $1-\frac{\gamma}{t}\geq\beta^t\geq 1-\frac{1}{t}$ for some $\gamma\in(0,1]$ and $\sqrt{t}\epsilon^t\geq\sqrt{t-1}\epsilon^{t-1}$. We first bound T_1. Based on the relations between v^t and v^{t-1} and $\beta^t\geq 1-\frac{1}{t}$, we can prove for any t,j

$$\sqrt{t}(A_j^t)=\sqrt{t}\left(\sqrt{v_j^t}+\epsilon^t\right)=\sqrt{t}\left(\sqrt{\beta^t v_j^{t-1}+(1-\beta^t)(\hat{g}_j^t)^2}+\epsilon^t\right)\geq\sqrt{(t-1)v_j^{t-1}}+\sqrt{t-1}\epsilon^{t-1}. \tag{18}$$

So for any j,t,

$$\sqrt{t}A_j^t\geq\sqrt{t-1}A_j^{t-1}. \tag{19}$$

Hence,

$$\mathbb{E}\left[\sum_{t=2}^{T}\|w^t-w^*\|^2_{\sqrt{t}A^t-\sqrt{t-1}A^{t-1}}\right]=\mathbb{E}\left[\sum_{t=2}^{T}\sum_{j=1}^{d}(w_j^t-w_j^*)^2\left(\sqrt{tv_j^t}+\sqrt{t}\epsilon^t-\sqrt{(t-1)v_j^{t-1}}-\sqrt{t-1}\epsilon^{t-1}\right)\right] \tag{20}$$

$$=\mathbb{E}\left[\sum_{j=1}^{d}\sum_{t=2}^{T}(w_j^t-w_j^*)^2\left(\sqrt{tv_j^t}+\sqrt{t}\epsilon^t-\sqrt{(t-1)v_j^{t-1}}-\sqrt{t-1}\epsilon^{t-1}\right)\right] \tag{21}$$

$$\leq\mathbb{E}\left[\sum_{j=1}^{d}D^2\sum_{t=2}^{T}\left(\sqrt{tv_j^t}+\sqrt{t}\epsilon^t-\sqrt{(t-1)v_j^{t-1}}-\sqrt{t-1}\epsilon^{t-1}\right)\right] \tag{22}$$

$$=\mathbb{E}\left[\sum_{j=1}^{d}D^2\left(\sqrt{Tv_j^T}+\sqrt{T}\epsilon^T-\sqrt{v_j^1}-\epsilon^1\right)\right]. \tag{23}$$

We next bound T_2. We prove a variant of Lemma 4.1 in Mukkamala & Hein (2017). The major differences are in that we consider the stochastic case and estimating v^t on public data. We prove the following inequality by induction:

$$\sum_{t=1}^{T}\mathbb{E}\left[\frac{(g_j^t)^2}{\sqrt{tv_j^t}+\sqrt{t}\epsilon^t}\right]\leq\frac{2(2-\gamma)}{a\gamma}\mathbb{E}\left[\sqrt{Tv_j^T}+\sqrt{T}\epsilon^T\right],\ j\in[d]. \tag{24}$$

For $t=1$,

$$\mathbb{E}\left[\frac{(g_j^1)^2}{\sqrt{v_j^t}+\epsilon^1}\right]=\mathbb{E}\left[\frac{(g_j^1)^2}{\sqrt{(1-\beta^1)(\hat{g}_j^1)^2}+\epsilon^1}\right]\leq\mathbb{E}\left[\frac{(\hat{g}_j^1)^2}{a\sqrt{(1-\beta^1)(\hat{g}_j^1)^2}+\epsilon^1}\right]\leq\mathbb{E}\left[\frac{\sqrt{(1-\beta^1)(\hat{g}_j^1)^2}+\epsilon^1}{a(1-\beta^1)}\right]. \tag{25}$$

Suppose that the conclusion holds when $t=T-1$, i.e., for any $j\in[d]$,

$$\sum_{t=1}^{T-1}\mathbb{E}\left[\frac{(g_j^t)^2}{\sqrt{tv_j^t}+\sqrt{t}\epsilon^t}\right]\leq\frac{2(2-\gamma)}{a\gamma}\mathbb{E}\left[\sqrt{(T-1)v_j^{T-1}}+\sqrt{T-1}\epsilon^{T-1}\right]. \tag{26}$$

In addition, combined with the fact that $v_j^T=\beta^T v_j^{T-1}+(1-\beta^T)(\hat{g}_j^T)^2$ and $\sqrt{T}\epsilon^T\geq\sqrt{T-1}\epsilon^{T-1}$, we have

$$\sqrt{(T-1)v_j^{T-1}}+\sqrt{T-1}\epsilon^{T-1}\leq\sqrt{\frac{(T-1)v_j^T}{\beta^T}-\frac{(T-1)(1-\beta^T)(\hat{g}_j^T)^2}{\beta^T}}+\sqrt{T}\epsilon^T \tag{27}$$

$$\leq\sqrt{Tv_j^T-\frac{(T-1)(1-\beta^T)(\hat{g}_j^T)^2}{\beta^T}}+\sqrt{T}\epsilon^T \tag{28}$$

$$\leq\sqrt{Tv_j^T}-\frac{(T-1)(1-\beta^T)(\hat{g}_j^T)^2}{2\beta^T\left(\sqrt{Tv_j^T}+\sqrt{T}\epsilon^T\right)}+\sqrt{T}\epsilon^T \tag{29}$$

$$\leq\sqrt{Tv_j^T}+\sqrt{T}\epsilon^T-\frac{a(T-1)(1-\beta^T)(g_j^T)^2}{2\beta^T\left(\sqrt{Tv_j^T}+\sqrt{T}\epsilon^T\right)}. \tag{30}$$

The third inequality comes from $\sqrt{a-b}\leq\sqrt{a}-\frac{b}{2\sqrt{a}}$ $(a\geq b)$ by letting a be Tv_j^T and b be $\frac{(T-1)(1-\beta^T)(\hat{g}_j^T)^2}{\beta^T}$. Hence

$$\sum_{t=1}^{T}\mathbb{E}\left[\frac{(g_j^t)^2}{\sqrt{tv_j^t+\sqrt{t}\epsilon^t}}\right]\leq\frac{2(2-\gamma)}{a\gamma}\mathbb{E}\left[\sqrt{Tv_j^T}+\sqrt{T}\epsilon^T-\frac{a(T-1)(1-\beta^T)(g_j^T)^2}{2\beta^T\left(\sqrt{Tv_j^T}+\sqrt{T}\epsilon^T\right)}\right]+\mathbb{E}\left[\frac{(g_j^T)^2}{\sqrt{Tv_j^T}+\sqrt{T}\epsilon^T}\right] \tag{31}$$

$$\leq\frac{2(2-\gamma)}{a\gamma}\mathbb{E}\left[\sqrt{Tv_j^T}+\sqrt{T}\epsilon^T\right]+\left(1-\frac{(2-\gamma)(T-1)(1-\beta^T)}{\gamma\beta^T}\right)\mathbb{E}\left[\frac{(g_j^T)^2}{\sqrt{Tv_j^T}+\sqrt{T}\epsilon^T}\right] \tag{32}$$

$$\leq\frac{2(2-\gamma)}{a\gamma}\mathbb{E}\left[\sqrt{Tv_j^T}+\sqrt{T}\epsilon^T\right]. \tag{33}$$

We then bound T_2 as follows.

$$T_2=\mathbb{E}\left[\sum_{t=1}^{T}\frac{\alpha^t}{2}\sum_{j=1}^{d}\frac{(g_j^t)^2}{\sqrt{v_j^t+\epsilon^t}}\right]=\frac{\alpha}{2}\mathbb{E}\left[\sum_{t=1}^{T}\sum_{j=1}^{d}\frac{(g_j^t)^2}{\sqrt{tv_j^t+\sqrt{t}\epsilon^t}}\right]\leq\frac{\alpha}{2}\sum_{j=1}^{d}\frac{2(2-\gamma)}{a\gamma}\mathbb{E}\left[\sqrt{Tv_j^T}+\sqrt{T}\epsilon^T\right]. \tag{34}$$

Noting that

$$\frac{\left\|w^1-w^*\right\|_{A^1}^2}{2\alpha}\leq\frac{D^2}{2\alpha}\sum_{j=1}^{d}\left(\sqrt{v_j^1}+\epsilon^1\right), \tag{35}$$

combined with the bounds of T_1,T_2 yields

$$\sum_{t=1}^{T}\left(\mathbb{E}[F(w^t)-F(w^*)]\right)\leq\left(\frac{D^2}{2\alpha}+\frac{\alpha(2-\gamma)}{a\gamma}\right)\sum_{j=1}^{d}\mathbb{E}\left[\sqrt{Tv_j^T}+\sqrt{T}\epsilon^T\right], \tag{36}$$

which implies that

$$\min_{t\in[T]}\mathbb{E}[F(w^t)]-F(w^*)\leq\left(\frac{D^2}{2\alpha}+\frac{\alpha(2-\gamma)}{a\gamma}\right)\frac{1}{T}\sum_{j=1}^{d}\mathbb{E}\left[\sqrt{Tv_j^T}+\sqrt{T}\epsilon^T\right]+\frac{1}{T}\sum_{t=1}^{T}\frac{\alpha}{2\sqrt{t}}\mathbb{E}\left[\|N\|_{A^t}^2\right] \tag{37}$$

$$\leq\left(\frac{D^2}{2\alpha}+\frac{\alpha(2-\gamma)}{a\gamma}\right)\frac{1}{\sqrt{T}}\sum_{j=1}^{d}\mathbb{E}\left[\sqrt{v_j^T}+\epsilon^T\right]+\frac{\alpha}{\sqrt{T}}\max_{t\in[T]}\mathbb{E}\left[\|N\|_{A^t}^2\right]. \tag{38}$$

The first term is standard for the RMSprop optimizer, and the last term is due to noise added to ensure differential privacy. To guarantee overall (ε,δ)-differential privacy by running T total iterations, we set $\sigma^2=O\left(\frac{b^2T\log(1/\delta)}{n^2\varepsilon^2}\right)$ and $T=O\left(\frac{n^2\varepsilon^2}{\max_{t\in[T]}\mathbb{E}\left[\sum_{j=1}^{d}A_j^t\right]\log(1/\delta)}\right)$. The convergence rate becomes

$$\min_{t\in[T]}\mathbb{E}[F(w^t)]-F(w^*)\leq O\left(\frac{\sqrt{\max_{t\in[T]}\mathbb{E}\left[\sum_{j=1}^{d}A_j^t\right]\log(1/\delta)}}{n\varepsilon}\right). \tag{39}$$

A.1.1. Proof for Theorem 3 (Fix A^t before training)

Denote the side information as a fixed A at any iteration t. Similar as previous analysis, setting a decaying learning rate $\alpha^t=\frac{\alpha}{\sqrt{t}}$, we have

$$\sum_{t=1}^{T}\left(\mathbb{E}[F(w^t)]-F(w^*)\right) \tag{40}$$

$$\leq\frac{\|w^1-w^*\|_A^2}{2\alpha}+\underbrace{\sum_{t=2}^{T}\frac{\mathbb{E}\left[\|w^t-w^*\|_{\sqrt{t}A-\sqrt{t-1}A}^2\right]}{2\alpha}}_{T_1}+\underbrace{\sum_{t=1}^{T}\frac{\alpha}{2\sqrt{t}}\mathbb{E}\left[\left\langle g^t,(A)^{-1}g^t\right\rangle\right]}_{T_2}+\sum_{t=1}^{T}\frac{\alpha}{2\sqrt{t}}\mathbb{E}\left[\|N\|_A^2\right]. \tag{41}$$

To bound T_1, we have

$$\sum_{t=2}^{T}\mathbb{E}\left[\|w^t-w^*\|^2_{\sqrt{t}A-\sqrt{t-1}A}\right]=\mathbb{E}\left[\sum_{t=2}^{T}\sum_{j=1}^{d}(w_j^t-w_j^*)^2\left((\sqrt{t}-\sqrt{t-1})(A_j)\right)\right]\leq O\left(\sqrt{T}\sum_{j=1}^{d}A_j\right)-\left\|w^1-w^*\right\|_A^2. \quad (42)$$

We consider T_2 next. From the assumptions on the clipping bound,

$$R:=\max_{j,t}\frac{\mathbb{E}\left[(g_j^t)^2\right]}{A_j^2}\leq C^2. \quad (43)$$

Then

$$\sum_{t=1}^{T}\sum_{j=1}^{d}\frac{\alpha}{2\sqrt{t}A_j}\mathbb{E}\left[(g_j^t)^2\right]\leq\sum_{t=1}^{T}\sum_{j=1}^{d}\frac{\alpha}{2\sqrt{t}}RA_j\leq\sqrt{T}\alpha R\sum_{j=1}^{d}A_j. \quad (44)$$

Therefore, we obtain

$$\min_{t\in[T]}\mathbb{E}[F(w^t)]-F(w^*)\leq O\left(\frac{1}{\sqrt{T}}\sum_{j=1}^{d}A_j+\frac{\alpha R}{\sqrt{T}}\sum_{j=1}^{d}A_j+\frac{\alpha}{\sqrt{T}}\mathbb{E}\left[\|N\|_A^2\right]\right). \quad (45)$$

A.2. Proof for Theorem 4 (Non-convex and smooth cases)

We use the same ϵ at each iteration. Let $\nabla_j F(w)$ denote the j-th coordinate of $\nabla F(w)$ for any w. Based on the L-smoothness of $F(\cdot)$,

$$F(w^{t+1})\leq F(w^t)-\alpha^t\sum_{j=1}^{d}\left(\nabla_j F(w^t)\cdot\left(\frac{g_j^t}{\sqrt{v_j^t+\epsilon}}+N\right)\right)+\frac{(\alpha^t)^2L}{2}\sum_{j=1}^{d}\left(\frac{(g_j^t)^2}{\left(\sqrt{v_j^t+\epsilon}\right)^2}+N^2+\frac{g_j^t}{\sqrt{v_j^t+\epsilon}}\cdot 2N\right), \quad (46)$$

where $N\sim\frac{1}{b}\mathcal{N}(0,\sigma^2C^2)$ and $\mathbb{E}[N^2]=\frac{\sigma^2C^2}{b^2}$. Taking expectation conditioned on w^t on both sides gives

$$\mathbb{E}_t[F(w^{t+1})]\leq F(w^t)-\alpha^t\sum_{j=1}^{d}\nabla_j F(w^t)\mathbb{E}_t\left[\frac{g_j^t}{\sqrt{v_j^t+\epsilon}}\right]+\frac{(\alpha^t)^2L}{2}\sum_{j=1}^{d}\mathbb{E}_t\left[\frac{(g_j^t)^2}{\left(\sqrt{v_j^t+\epsilon}\right)^2}\right]+\frac{(\alpha^t)^2Ld}{2b^2}\sigma^2C^2. \quad (47)$$

The following proof is extended from that of Theorem 1 in Reddi et al. (2018a).

$$\begin{aligned}\mathbb{E}_t[F(w^{t+1})]&\leq F(w^t)-\alpha^t\sum_{j=1}^{d}\nabla_j F(w^t)\mathbb{E}_t\left[\frac{g_j^t}{\sqrt{v_j^t+\epsilon}}-\frac{g_j^t}{\sqrt{\beta v_j^{t-1}+\epsilon}}+\frac{g_j^t}{\sqrt{\beta v_j^{t-1}+\epsilon}}\right]\\&\quad+\frac{(\alpha^t)^2L}{2}\sum_{j=1}^{d}\mathbb{E}_t\left[\frac{(g_j^t)^2}{\left(\sqrt{v_j^t+\epsilon}\right)^2}\right]+\frac{(\alpha^t)^2Ld}{2b^2}\sigma^2C^2 \quad (48)\\&\leq F(w^t)-\alpha^t\sum_{j=1}^{d}\frac{(\nabla_j F(w^t))^2}{\sqrt{\beta v_j^{t-1}+\epsilon}}+\alpha^t\sum_{j=1}^{d}\left|\nabla_j F(w^t)\right|\left|\mathbb{E}_t\left[\frac{g_j^t}{\sqrt{v_j^t+\epsilon}}-\frac{g_j^t}{\sqrt{\beta v_j^{t-1}+\epsilon}}\right]\right|\\&\quad+\frac{(\alpha^t)^2L}{2}\sum_{j=1}^{d}\mathbb{E}_t\left[\frac{(g_j^t)^2}{\left(\sqrt{v_j^t+\epsilon}\right)^2}\right]+\frac{(\alpha^t)^2Ld}{2b^2}\sigma^2C^2. \quad (49)\end{aligned}$$

Further,

$$\frac{g_j^t}{\sqrt{v_j^t}+\epsilon}-\frac{g_j^t}{\sqrt{\beta v_j^{t-1}}+\epsilon}\leq\left|g_j^t\right|\left|\frac{1}{\sqrt{v_j^t}+\epsilon}-\frac{1}{\sqrt{\beta v_j^{t-1}}+\epsilon}\right| \tag{50}$$

$$=\frac{\left|g_j^t\right|}{\left(\sqrt{v_j^t}+\epsilon\right)\left(\sqrt{\beta v_j^{t-1}}+\epsilon\right)}\frac{(1-\beta)(\hat{g}_j^t)^2}{\left(\sqrt{v_j^t}+\sqrt{\beta v_j^{t-1}}\right)} \tag{51}$$

$$=\frac{\left|g_j^t\right|}{\left(\sqrt{v_j^t}+\epsilon\right)\left(\sqrt{\beta v_j^{t-1}}+\epsilon\right)}\frac{(1-\beta)(\hat{g}_j^t)^2}{\left(\sqrt{\beta v_j^{t-1}+(1-\beta)(\hat{g}_j^t)^2}+\sqrt{\beta v_j^{t-1}}\right)} \tag{52}$$

$$\leq\frac{1}{\left(\sqrt{v_j^t}+\epsilon\right)\left(\sqrt{\beta v_j^{t-1}}+\epsilon\right)}\frac{\sqrt{1-\beta}}{\sqrt{a}}(g_j^t)^2\leq\frac{1}{\left(\sqrt{\beta v_j^{t-1}}+\epsilon\right)\epsilon}\frac{\sqrt{1-\beta}}{\sqrt{a}}(g_j^t)^2. \tag{53}$$

We have used the observation $\frac{(1-\beta)(\hat{g}_j^t)^2}{\left(\sqrt{\beta v_j^{t-1}+(1-\beta)(\hat{g}_j^t)^2}+\sqrt{\beta v_j^{t-1}}\right)}\leq\sqrt{1-\beta}\hat{g}_j^t$, and $\hat{g}_j^t\leq\frac{1}{\sqrt{a}}g_j^t$.

From L-smoothness of $F(\cdot)$ which implies that $\|\nabla F(u)-\nabla F(v)\|\leq L\|u-v\|$ for any $u,v\in\mathbb{R}^d$, and Assumption 1, it is easy to see there exists a constant B such that $|\nabla_j F(w)|\leq B$ for any $j\in[d]$.

$$\mathbb{E}_t[F(w^{t+1})] \tag{54}$$

$$\leq F(w^t)-\alpha^t\sum_{j=1}^d\frac{(\nabla_j F(w^t))^2}{\sqrt{\beta v_j^{t-1}}+\epsilon}+\frac{\alpha^t B\sqrt{1-\beta}}{\epsilon\sqrt{a}}\sum_{j=1}^d\mathbb{E}_t\left[\frac{(g_j^t)^2}{\sqrt{\beta v_j^{t-1}}+\epsilon}\right]+\frac{(\alpha^t)^2Ld}{2}\sum_{j=1}^d\mathbb{E}_t\left[\frac{(g_j^t)^2}{\left(\sqrt{v_j^t}+\epsilon\right)^2}\right]+\frac{(\alpha^t)^2Ld}{2b^2}\sigma^2C^2 \tag{55}$$

$$\leq F(w^t)-\alpha^t\sum_{j=1}^d\frac{(\nabla_j F(w^t))^2}{\sqrt{\beta v_j^{t-1}}+\epsilon}+\frac{\alpha^t B\sqrt{1-\beta}}{\epsilon\sqrt{a}}\sum_{j=1}^d\mathbb{E}_t\left[\frac{(g_j^t)^2}{\sqrt{\beta v_j^{t-1}}+\epsilon}\right]+\frac{(\alpha^t)^2L}{2\epsilon}\sum_{j=1}^d\mathbb{E}_t\left[\frac{(g_j^t)^2}{\sqrt{\beta v_j^{t-1}}+\epsilon}\right]+\frac{(\alpha^t)^2Ld}{2b^2}\sigma^2C^2, \tag{56}$$

where the last inequality holds due to $\left(\sqrt{v_j^t}+\epsilon\right)^2\geq\epsilon\left(\epsilon+\sqrt{v_j^t}\right)\geq\epsilon\left(\epsilon+\sqrt{\beta v_j^{t-1}}\right)$. Lemma 1 in Reddi et al. (2018a) proves that $\mathbb{E}_t\left[(g_j^t)^2\right]\leq\frac{\tau_j^2}{b}+(\nabla_j F(w^t))^2$ where τ_j^2 is the variance bound of the j-th coordinate, i.e., $\mathbb{E}\left[\|g_j^t-\nabla_j F(w^t)\|^2\right]\leq\tau_j^2$. Plugging this inequality into Eq. (56), combined with $\frac{L\alpha^t}{2\epsilon}\leq\frac{1}{4}$ and $\frac{B\sqrt{1-\beta}}{\sqrt{a}\epsilon}\leq\frac{1}{4}$, we obtain

$$\mathbb{E}_t[F(w^{t+1})]\leq F(w^t)-\frac{\alpha^t}{2(\sqrt{\beta}B+\epsilon)}\|\nabla F(w^t)\|^2+\left(\frac{\alpha^t B\sqrt{1-\beta}}{\sqrt{a}\epsilon^2}+\frac{L(\alpha^t)^2}{2\epsilon^2\sqrt{\beta}}\right)\frac{\sum_{j\in[d]}\tau_j^2}{b}+\frac{(\alpha^t)^2Ld}{2b^2}\sigma^2C^2. \tag{57}$$

Taking expectation on both sides and applying the telescope sum yields

$$\frac{1}{T}\sum_{t\in[T]}\mathbb{E}[\|\nabla F(w^t)\|^2]\leq O\left(\frac{1}{T}\right)+O\left(\frac{\|\tau^2\|_1}{b}\right)+O\left(\frac{\sigma^2}{b^2}\right) \tag{58}$$

A.2.1. Proof for Theorem 5 (Fix A before training)

Due to L-smoothness of $F(\cdot)$, we have

$$F(w^{t+1})\leq F(w^t)-\alpha^t\sum_{j=1}^d\left(\nabla_j F(w^t)\cdot\left(\frac{g_j^t}{A_j}+N\right)\right)+\frac{(\alpha^t)^2L}{2}\sum_{j=1}^d\left(\frac{(g_j^t)^2}{A_j^2}+N^2+\frac{g_j^t}{A_j}\cdot 2N\right), \tag{59}$$

where $N \sim \frac{1}{b}\mathcal{N}(0,\sigma^2 C^2)$. Taking expectation conditioned on w^t on both sides gives

$$\mathbb{E}_t\big[F(w^{t+1})\big] \leq F(w^t) - \alpha^t \sum_{j=1}^{d} \frac{(\nabla_j F(w^t))^2}{A_j} + \frac{(\alpha^t)^2 L}{2} \sum_{j=1}^{d} \frac{1}{A_j^2} \mathbb{E}_t\big[(g_j^t)^2\big] + \frac{(\alpha^t)^2 L d}{2b^2}\sigma^2 C^2 \tag{60}$$

$$\leq F(w^t) - \alpha^t \sum_{j=1}^{d} \frac{(\nabla_j F(w^t))^2}{A_j} + \frac{(\alpha^t)^2 L}{2} \sum_{j=1}^{d} \frac{1}{A_j^2} \left(\frac{\tau_j^2}{b} + (\nabla_j F(w^t))^2\right) + \frac{(\alpha^t)^2 L d}{2b^2}\sigma^2 C^2 \tag{61}$$

$$\leq F(w^t) - \alpha^t \sum_{j=1}^{d} \frac{(\nabla_j F(w^t))^2}{A_j} + \frac{(\alpha^t)^2 L}{2\epsilon} \sum_{j=1}^{d} \frac{1}{A_j} \left(\frac{\tau_j^2}{b} + (\nabla_j F(w^t))^2\right) + \frac{(\alpha^t)^2 L d}{2b^2}\sigma^2 C^2 \tag{62}$$

$$\leq F(w^t) - \frac{\alpha^t}{2} \big\|\nabla F(w^t)\big\|_{A^{-1}}^2 + \frac{(\alpha^t)^2 L}{2\epsilon} \sum_{j=1}^{d} \frac{\tau_j^2}{A_j b} + \frac{(\alpha^t)^2 L d}{2b^2}\sigma^2 C^2. \tag{63}$$

The last inequality is due to $\alpha^t \leq \frac{\epsilon}{L}$. Taking expectation on both sides yields

$$\mathbb{E}\big[F(w^{t+1})\big] \leq \mathbb{E}\big[F(w^t)\big] - \frac{\alpha^t}{2}\mathbb{E}[\|\nabla F(w^t)\|_{A^{-1}}^2] + \frac{(\alpha^t)^2 L}{2\epsilon} \sum_{j=1}^{d} \frac{\tau_j^2}{A_j b} + \frac{(\alpha^t)^2 L d}{2b^2}\sigma^2 C^2. \tag{64}$$

Similarly, by rearranging terms and applying telescope sum, we obtain

$$\frac{1}{T}\sum_{t\in[T]} \mathbb{E}\big[\|\nabla F(w^t)\|_{A^{-1}}^2\big] \leq O\left(\frac{1}{T}\right) + O\left(\frac{\tau^2}{A}\cdot\frac{1}{b}\right) + O\left(\frac{\sigma^2}{b^2}\right). \tag{65}$$

B. Experimental Details

Pseudo Code of some Algorithms. For completeness, we present the full baseline DP-Adam algorithm in Algorithm 2 and `AdaDPS` extended to federated learning in Algorithm 3.

Algorithm 2: DP-Adam (Zhou et al., 2020)

Input: T, batch size b, noise multiplier σ, clipping threshold C, initial model $w^1 \in \mathbb{R}^d$, $v^0 = \mathbf{0}$, $m^0 = \mathbf{0}$, small constant vector $\epsilon^t \in \mathbb{R}^d$, learning rate α^t, moving average parameters β_1, β_2

for $t = 1, \cdots, T-1$ **do**

Uniformly randomly sample a mini-batch B with size b from private training data

Get individual gradients for sample $i \in B$:

$$g^{i,t} \leftarrow \nabla f(x^i; w^t)$$

Private gradients using Gaussian mechanism:

$$\tilde{g}^t \leftarrow \frac{1}{b}\sum_{i\in B} \text{clip}\big(g^{i,t}, C\big) + \frac{1}{b}\mathcal{N}(0, \sigma^2 C^2)$$

Update first and second moment estimates as

$$m^t \leftarrow \beta_1 m^{t-1} + (1-\beta_1)\tilde{g}^t$$
$$v^t \leftarrow \beta_2 v^{t-1} + (1-\beta_2)(\tilde{g}^t)^2$$

Update the model parameter w as

$$w^{t+1} \leftarrow w^t - \alpha^t \frac{m^t/(1-\beta_1^t)}{\sqrt{v^t/(1-\beta_2^t)} + \epsilon^t},$$

where β_1^t, β_2^t denotes β_1, β_2 to the power of t (with slight abuse of notations)

return w^T

Algorithm 3: `AdaDPS` applied to federated learning

Input: T communication rounds, b selected devices each round, noise multiplier σ, clipping threshold C, initial model $w^1 \in \mathbb{R}^d$, non-sensitive side information A, number of local iterations s, local learning rate η^t

for $t=1,\cdots,T-1$ **do**

Server uniformly selects a subset B of b devices and sends the current global model w^t to them

Each device $i \in B$ sets the local model to be the current global model:

$$w^{i,0} \leftarrow w^t$$

Each device $i \in B$ runs adaptive mini-batch SGD locally with side information A to obtain model updates:

for $j=0,\cdots,s$ **do**

$$w^{i,j+1} \leftarrow w^{i,j} - \eta^t \frac{\nabla f(w^{i,j})}{A}$$

And then privatize model updates:

$$\Delta^{i,t} \leftarrow w^{i,s+1} - w^{i,0}$$

$$\tilde{\Delta}^{i,t} \leftarrow \text{clip}(\Delta^{i,t}, C) + \mathcal{N}(0, \sigma^2 C^2)$$

Each device $i \in B$ sends $\tilde{\Delta}^{i,t}$ to the server

Server updates the global model as:

$$w^{t+1} \leftarrow w^t + \frac{1}{b} \sum_{i \in B} \tilde{\Delta}^{i,t}$$

return w^T

Datasets and Models. We consider a diverse set of datasets and tasks.

- **StackOverflow** (Authors, 2019) consists of posts on the Stack Overflow website, where the task is tag prediction (500-class classification). We randomly subsample 246,092 samples from the entire set. There are 10,000 input features in StackOverflow, resulting in more than 5 million learnable parameters in a logistic regression model.
- **IMDB** (Maas et al., 2011) is widely used for for binary sentiment classification of movie reviews, consisting of 25,000 training and 25,000 testing samples. We study two models on IMDB: logistic regression and neural networks (with LSTM) with 20,002 and 706,690 parameters, respectively. For logistic regression, we set the vocabulary size to 10,000 and consider two sets of commonly-used features separately: bag-of-words (BoW) and TF-IDF values.
- **MNIST** (LeCun et al., 1998) images with a deep autoencoder model (for image reconstruction) which has the same architecture as that in previous works (Reddi et al., 2018a) (containing more than 2 million parameters). The loss is reconstruction error measured as the mean squared distance in the pixel space. We scale each input feature to the range of $[0,1]$.

Hyperparameter Tuning. We detail our hyperparameter tuning protocol and the hyperparameter values here. Our code is publicly available at `github.com/litian96/AdaDPS`.

- For non-private training experiments, we fix the mini-batch size to 64, and tune fixed learning rates by performing a grid search over $\{0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.2, 0.5, 1, 2\}$ separately for all methods on validation data. We do not use momentum for AdaS (i.e., applying the idea of preconditioning of `AdaDPS` without privatization) for all centralized training experiments.
- For differentially private training, the δ values in the privacy budget are always inverse of the number of training samples. We fix the noise multiplier σ for each dataset, tune the clipping threshold, and compute the final ε values. Specifically, the σ values are 1, 1, and 0.95 for IMDB (convex), IMDB (LSTM), and StackOverflow; 1 and 0.75 for MNIST (autoencoder). The clipping threshold C (in Algorithm 1) is tuned from $\{0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 3\}$, jointly with tuning the (fixed) learning rates. The number of micro-batches is 16 for all related experiments, and the mini-batch size is 64 (i.e., we privatize each gradient averaged over 4 individual ones to speed up computation).

- For federated learning experiments, we fix server-side learning rate to be 1 (i.e., simply applying the unscaled average of noisy model updates in Line 9 of Algorithm 3), and apply server-side momentum (Reddi et al., 2021) with a moving average parameter 0.9 for all methods in the left sub-figure in Figure 6. The number of local epochs is set to 1 for all runs, and the local mini-batch size is 100.

The tuned hyperparameter values (clipping threshold C, learning rate) for private training are summarized in Table 5 below.

Datasets	**DP-SGD**	**DP-Adam**	**DP-RMSProp**	**AdaDPS (RMSProp)**	**AdaDPS (Adam)**
IMDB (convex)	(0.1, 1)	(0.02, 0.1)	(0.05, 0.1)	(2, 0.5)	(2, 1)
IMDB (LSTM)	(0.1, 0.1)	(0.1, 0.001)	(0.1, 0.001)	(0.2, 0.1)	(0.2, 0.05)
StackOverflow (linear)	(0.1, 1)	(0.1, 0.01)	(0.2, 0.01)	(1, 0.5)	(1, 0.5)
MNIST (autoencoder)	(0.05, 0.5)	(0.01, 0.001)	(0.01, 0.001)	(3, 0.005)	(3, 0.005)
MNIST (classification)	(0.5, 0.01)	(0.5, 0.001)	(0.5, 0.001)	(2, 0.005)	(2, 0.005)

Table 5. Major hyperparameter values (learning rate and clipping threshold C) used in private experiments for all datasets. 'IMDB (convex)' is IMDB (BoW features) on a logistic regression model. StackOverflow results are for centralized training. The noise multiplier σ values in these four tasks are 1, 1, 0.95, 1, respectively, resulting in ε values being 1.5, 2.8, 0.84, and 1.6.

C. Additional Results

C.1. Additional Baselines

Other Private Adaptive Optimization Baselines. In the main text, we mainly compare AdaDPS with DP-Adam (summarized in Algorithm 2). There are other possible baselines similar as DP-Adam, by replacing Adam with other adaptive methods, resulting in DP-AdaGrad and DP-RMSProp. This line of differentially private optimizers has similar empirical performance as DP-Adam, as shown in the results in Table 6 below (using DP-RMSProp as an example).

Datasets	**Metrics**	**DP-SGD**	**DP-Adam**	**DP-RMSProp**	**AdaDPS (RMSProp)**	**AdaDPS (Adam)**
IMDB (convex)	accuracy	0.63	0.69	0.67	0.80	0.80
IMDB (LSTM)	accuracy	0.70	0.69	0.69	0.73	0.73
StackOverflow (linear)	accuracy	0.28	0.30	0.31	0.40	0.40
MNIST (autoencoder)	loss ($\times$100)	5.013	3.697	3.636	3.566	3.443
MNIST (classification)	accuracy	0.9273	0.9333	0.9314	0.9377	0.9541

Table 6. Full comparisons between AdaDPS and private adaptive optimization methods. The evaluation metrics are reported on test data. 'IMDB (convex)' is IMDB (BoW features) on a logistic regression model. For (ε,δ)-differential privacy, the ε values of experiments in the five rows are 1.5, 2.8, 0.84, 1.6, and 1.25, respectively, and the δ values are the inverse of the number of training samples (as mentioned in the main text).

Using Public Data for Pretraining. Another possible way of leveraging public data to improve privacy/utility tradeoffs is to pretrain on them. However, this would give only limited performance improvement if the amount of public data is very small. In the main text, when needed, AdaDPS randomly samples 1% training data as public data. Under this setup, we empirically compare AdaDPS with the pre-training baseline (denoted as DP-SGD w/ warm start). From Table 7, we see that AdaDPS outperforms it by a large margin.

Datasets	**Metrics**	**DP-SGD**	**DP-SGD** w/ warm start	**AdaDPS** w/ public
IMDB (convex)	accuracy	0.63	0.73	0.80
StackOverflow	accuracy	0.28	0.33	0.40
MNIST (autoencoder)	loss	0.050	0.049	0.036

Table 7. Compare AdaDPS with an additional baseline of DP-SGD pre-trained on public data on three datasets. For 'DP-SGD w/ warm start', we first train on public data for 10 epochs via adaptive methods (RMSProp), and then run DP-SGD on private data starting from that initialization.

DP-Adam with Public Data. In the main text (Section 6.1.1), we discuss the DP-R-Pub. baseline based on the RMSProp method with the preconditioner estimated on public data. Similarly, one can also apply such clean preconditioners in DP-Adam updates, resulting in another baseline which we call DP-Adam-Pub. The main differences between DP-Adam-Pub and DP-R-Pub are that DP-Adam-Pub additionally considers momentum. Formally, the updates of w^t is as follows:

$$\tilde{g}^t \leftarrow \frac{1}{b}\sum_{i\in B}\text{clip}\big(g^{i,t},C\big)+\frac{1}{b}\mathcal{N}(0,\sigma^2C^2), \quad \hat{g}^t \leftarrow \mathbb{E}_x\left[\nabla f(x;w^t)\right] \text{ for } x\in x_{\text{pub}},$$
$$m^t \leftarrow \beta_2 m^t+(1-\beta_2)\big(\beta_1\tilde{g}^t+(1-\beta_1)\hat{g}^t\big), \quad v^t \leftarrow \beta_3 v^t+(1-\beta_3)(\hat{g}^t)^2,$$
$$w^{t+1} \leftarrow w^t-\alpha^t\frac{m^t/(1-(\beta_2)^t)}{\sqrt{v^t/(1-(\beta_3)^t)}+\epsilon},$$

where $\beta_1,\beta_2,\beta_3,\epsilon$ are small constants.

Datasets	**Metrics**	**DP-SGD**	**DP-Adam-Pub**	**AdaDPS** w/ public
IMDB (convex)	accuracy	0.63	0.74	0.80
StackOverflow	accuracy	0.28	0.31	0.40
MNIST (autoencoder)	loss	0.050	0.064	0.036

Table 8. Results of comparing AdaDPS with to DP-Adam-Pub (i.e., DP-Adam using clean preconditioners estimated on public data).

C.2. Side Information in Non-Private Training

In the main text, we mainly focus on private optimization. It is expected that side information (even without the assist of public data) would also be beneficial in non-private settings, which could serve as a simple alternative to adaptive methods. We report results in Table 9.

Datasets	**Metrics**	**SGD**	**Adam**	**AdaS** (w/ public)	**AdaS** (w/o public)
IMDB (convex)	accuracy	0.66	0.88	0.88	0.88
IMDB (LSTM)	accuracy	0.88	0.88	0.88	0.88
StackOverflow (linear)	accuracy	0.38	0.64	0.64	0.64
MNIST (autoencoder)	loss ($\times$100)	5.013	1.151	1.805	—

Table 9. Performance of each method in *non-private* training. We see that AdaS can match the performance of Adam in non-private settings.

C.3. Effects of Public Data Size

We further study the effects of public data size. Only a very small set of public data (even 0.04% the size of private training data) can provide good preconditioner estimates.

Datasets	*upper bound*	**AdaDPS** 1% public	**AdaDPS** 5$\times$ less	**AdaDPS** 25$\times$ less
IMDB (convex)	0.82	0.80	0.80	0.75
StackOverflow	0.41	0.40	0.40	0.39

Table 10. Effects of public data sizes.