Towards Better Generalization of Adaptive Gradient Methods

Anonymous Author(s)
Affiliation
Address
email

Abstract

Adaptive gradient methods such as AdaGrad, RMSprop and Adam have been optimizers of choice for deep learning due to their fast training speed. However, it was recently observed that their generalization performance is often worse than that of SGD for over-parameterized neural networks. While new algorithms such as AdaBound, SWAT, and Padam were proposed to improve the situation, the provided analyses are only committed to optimization bounds for the training objective, leaving critical generalization capacity unexplored. To close this gap, we propose ***S**table **A**daptive **G**radient **D**escent* (SAGD) for nonconvex optimization which leverages differential privacy to boost the generalization performance of adaptive gradient methods. Theoretical analyses show that SAGD has high-probability convergence to a population stationary point. We further conduct experiments on various popular deep learning tasks and models. Experimental results illustrate that SAGD is empirically competitive and often better than baselines.

1 Introduction

We consider in this paper, the following minimization problem:

$$\min_{\mathbf{w}\in\mathcal{W}} f(\mathbf{w}) \triangleq \mathbb{E}_{z\sim\mathcal{P}}[\ell(\mathbf{w}, z)] \;, \tag{1}$$

where the *population loss* f is a (possibly) nonconvex objective function (as for most deep learning tasks), $\mathcal{W} \subset \mathbb{R}^d$ is the parameter set and z is the vector of data samples distributed according to an unknown data distribution $\mathcal{P}$. We assume that we have access to an oracle that, given n i.i.d. samples $(\mathbf{z}_1, \ldots, \mathbf{z}_n)$, returns the stochastic objectives $(\ell(\mathbf{w}, \mathbf{z}_1), \ldots, \ell(\mathbf{w}, \mathbf{z}_n))$. Our goal is to find critical points of the population loss function (1). Given the unknown data distribution, a natural approach towards solving (1) is empirical risk minimization (ERM) [29], which minimizes the *empirical loss* $\hat{f}(\mathbf{w})$ as follows: $\min_{\mathbf{w}\in\mathcal{W}} \hat{f}(\mathbf{w}) \triangleq \frac{1}{n}\sum_{j=1}^{n} \ell(\mathbf{w}, \mathbf{z}_j)$, when n samples $\mathbf{z}_1, \ldots, \mathbf{z}_n$ are observed. Stochastic gradient descent (SGD) [28] which iteratively updates the parameter of a model by descending in the direction of the negative gradient, computed on a single sample or a mini-batch of samples, has been the most dominant algorithm for solving the ERM problem, e.g., training deep neural networks. To automatically tune the learning-rate decay in SGD, adaptive gradient methods, such as AdaGrad [6], RMSprop [31], and Adam [15], have emerged leveraging the curvature of the objective function resulting in adaptive coordinate-wise learning rates for faster convergence.

However, the generalization ability of these adaptive methods is often worse than that of SGD for over-parameterized neural networks, e.g., convolutional neural network (CNN) for image classification and recurrent neural network (RNN) for language modeling [35]. To mitigate this issue, several recent algorithms were proposed to combine adaptive methods with SGD. For example, AdaBound [20] and SWAT [14] switch from Adam to SGD as the training proceeds, while Padam [4, 37] unifies AMSGrad [27] and SGD with a partially adaptive parameter. Despite much efforts on deriving theoretical convergence results of the objective function [36, 34, 39, 5], these newly proposed adaptive gradient methods are often misunderstood regarding their generalization abilities,

Submitted to 34th Conference on Neural Information Processing Systems (NeurIPS 2020). Do not distribute.

which is the ultimate goal. On the other hand, current adaptive gradient methods [6, 15, 31, 27, 34] follow a typical stochastic optimization (SO) oracle paradigm [28, 11] which uses stochastic gradients to update the parameters. The SO oracle requires *new samples* at every iteration to get the stochastic gradient such that, in expectation, it equals the population gradient. In practice, however, only finite training samples are available and reused by the optimization oracle for a certain number of times (*i.e.,* epochs). Hardt et al. [12] found that the generalization error increases with the number of times the optimization oracle passes over the training data. It is thus expected that gradient descent algorithms will be much more well-behaved if we have access to an infinite number of fresh samples. Re-using data samples is therefore a caveat for the generalization of a given algorithm.

In order to tackle the above issues, we propose ***S****table* ***A****daptive* ***G****radient* ***D****escent* (SAGD) which aims at improving the generalization of general adaptive gradient descent algorithms. SAGD behaves similarly to the aforementioned ideal case of infinite fresh samples borrowing ideas from *adaptive data analysis* [8] and *differential privacy* [7]. The main idea of our method is that, at each iteration, SAGD accesses the observations z through a differentially private mechanism and computes an estimated gradient $\nabla \ell(\mathbf{w}, z)$ of the objective function $\nabla f(\mathbf{w})$. It then uses the estimated gradient to perform a descent step using adaptive stepsize. We prove that the reused data points in SAGD nearly possess the statistical nature of *fresh samples* yielding to high concentration bounds of the population gradients through the iterations. Our contributions can be summarized as follows:

- We derive a novel adaptive gradient method, namely SAGD, leveraging ideas of differential privacy and adaptive data analysis aiming at improving the generalization of current baseline methods. A mini-batch variant is also introduced for large-scale learning tasks.
- Our differentially private mechanism, embedded in the SAGD, explores the idea of Laplace Mechanism (adding Laplace noises to gradients) and THRESHOLDOUT [7] leading to DPG-LAP and DPG-SPARSE methods saving privacy cost. In particular, we show that differentially private gradients stay close to the population gradients with high probability.
- We establish various theoretical guarantees for our algorithm. We derive a concentration bound on the generalization error and show that the ℓ_2-norm of the *population gradient*, *i.e.,* $\|\nabla f(\mathbf{w})\|$ obtained by the SAGD converges in $\mathcal{O}(1/n^{2/3})$ with high probability.
- We conduct several experimental applications based on training neural networks for image classification and language modeling indicating that SAGD outperforms existing adaptive gradient methods in terms of the generalization and over-fitting performance.

Roadmap: The SAGD algorithm, including the differentially private mechanisms, and its mini-batch variant are described in Section 3. Numerical experiments are presented Section 4. Section 5 concludes our work. Due to space limit, most of the proofs are relegated to supplementary material.

Notations: We use $\mathbf{g}_t$ and $\nabla f(\mathbf{w})$ interchangeably to denote the *population gradient* such that $\mathbf{g}_t = \nabla f(\mathbf{w}_t) = \mathbb{E}_{\mathbf{z}\in\mathcal{P}}[\nabla \ell(\mathbf{w}_t, \mathbf{z})]$. $S = \{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$ denotes the n available training samples. $\hat{\mathbf{g}}_t$ denotes the sample gradient evaluated on S such that $\hat{\mathbf{g}}_t = \nabla \hat{f}(\mathbf{w}) = \frac{1}{n}\sum_{j=1}^{n} \nabla \ell(\mathbf{w}_t, \mathbf{z}_j)$. For a vector $\mathbf{v}$, $\mathbf{v}^2$ represents that $\mathbf{v}$ is element-wise squared. We use $\mathbf{v}^i$ or $[\mathbf{v}]_i$ to denote the i-th coordinate of $\mathbf{v}$ and $\|\mathbf{v}\|_2$ is the ℓ_2-norm of $\mathbf{v}$ and denote $[d] = \{1, \ldots, d\}$.

2 Preliminaries

Adaptive Gradient Methods: In the nonconvex setting, existing work on SGD [11] and adaptive gradient methods [36, 34, 39, 5] show convergence to a stationary point with a rate of $\mathcal{O}(1/\sqrt{T})$ where T is the number of stochastic gradient computations. Given n samples, a stochastic oracle can obtain at most n stochastic gradients, which implies convergence to the population stationarity with a rate of $\mathcal{O}(1/\sqrt{n})$. In addition, Kuzborskij and Lampert [17], Raginsky et al. [26], Hardt et al. [12], Mou et al. [23], Pensia et al. [24], Chen et al. [5], Li et al. [19] study the generalization of gradient-based optimization algorithms using the generalization property of stable algorithms [2]. In particular, Raginsky et al. [26], Mou et al. [23], Li et al. [19], Pensia et al. [24] focus on noisy gradient algorithms, e.g., SGLD, and provide a generalization bound in $\mathcal{O}(\sqrt{T}/n)$. This type of bounds usually has a dependence on the training data and has a polynomial dependence on T.

Differential Privacy and Adaptive Data Analysis: Differential privacy [7] was originally studied for preserving the privacy of individual data in the statistical query. Recently, differential privacy has

Algorithm 1 SAGD with DGP-LAP

```
1: Input: Dataset $S$, certain loss $\ell(\cdot)$, initial point $\mathbf{w}_0$ and noise level $\sigma$.
2: Set noise level $\sigma$, iteration number $T$, and stepsize $\eta_t$.
3: for $t = 0, ..., T-1$ do
4:     DPG-LAP: Compute full batch gradient on $S$:
                 $\hat{\mathbf{g}}_t = \frac{1}{n}\sum_{j=1}^{n}\nabla\ell(\mathbf{w}_t, z_j)$.
5:     Set $\tilde{\mathbf{g}}_t = \hat{\mathbf{g}}_t + \mathbf{b}_t$, where $\mathbf{b}_t^i$ is drawn i.i.d from Lap$(\sigma)$ for all $i \in [d]$.
6:     $\mathbf{m}_t = \tilde{\mathbf{g}}_t$ and $\mathbf{v}_t = (1-\beta_2)\sum_{i=1}^{t}\beta_2^{t-i}\tilde{\mathbf{g}}_i^2$.
7:     $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t\mathbf{m}_t/(\sqrt{\mathbf{v}_t} + \nu)$.
8: end for
```

been widely used for stochastic optimization. Some pioneering work [3, 1, 33] introduce differential privacy to empirical risk minimization (ERM) to protect sensitive information of the training data. The popular differentially private algorithms include the gradient perturbation that adds noise to the gradient in gradient descent algorithms [3, 1, 32]. Moreover, in Adaptive Data Analysis ADA [8, 9, 10], the same holdout set is used multiple times to test the hypotheses which are generated based on previous test results. It has been shown that reusing the holdout set via a differentially private mechanism ensures the validity of the test. In other words, the differentially private reused dataset maintains the statistical nature of fresh samples and improves generalization [38].

3 Stable Adaptive Gradient Descent Algorithm

Beforehand, we recall the definition of a (ϵ, δ)-differentially private algorithm:

Definition 1. *(Differential Privacy [7]) A randomized algorithm $\mathcal{M}$ is (ϵ, δ)-differentially private if*

$$\mathbb{P}\{\mathcal{M}(\mathcal{D}) \in \mathcal{Y}\} \leq \exp(\epsilon)\mathbb{P}\{\mathcal{M}(\mathcal{D}\prime) \in \mathcal{Y}\} + \delta$$

holds for all $\mathcal{Y} \subseteq Range(\mathcal{M})$ and all pairs of adjacent datasets $\mathcal{D}, \mathcal{D}\prime$ that differ on a single sample.

The general approach for achieving (ϵ, δ)-differential privacy when estimating a deterministic real-valued function $q : \mathcal{Z}^n \to \mathbb{R}^d$ is Laplace Mechanism [7], which adds Laplace noise calibrated to the function q, *i.e.*, $\mathcal{M}(\mathcal{D}) = q(\mathcal{D}) + \mathbf{b}$, where for all $i \in [d]$, $\mathbf{b}^i \sim$ Laplace$(0, \sigma^2)$. We present SAGD with two different **D**ifferential **P**rivate **G**radient (DPG) computing methods that provide an estimate of the gradient $\nabla f(\mathbf{w})$, namely DPG-LAP based on the *Laplace Mechanism* [7], see Section 3.1 and an improvement named DPG-SPARSE motivated by sparse vector technique [7] in Section 3.2.

3.1 SAGD with DGP-LAP

In most deep learning applications, a training set S of size n is observed. Then, at each iteration $t \in [T]$, SAGD, described in Algorithm 1, calls DPG-LAP (Line 5 in Algorithm 1), that computes the empirical gradient noted $\tilde{\mathbf{g}}_t$ and updates the model parameter $\mathbf{w}_{t+1}$ using adaptive stepsize. Note that the noise variance σ^2, step-size η_t, iteration number T, β_2 are parameters and play an important role for our theoretical study presented in the sequel. We first consider DPG-LAP which adds Laplace noise $\mathbf{b}_t \in \mathbb{R}^d$ to the empirical gradient $\hat{\mathbf{g}}_t = \frac{1}{n}\sum_{j=1}^{n}\nabla\ell(\mathbf{w}_t, \mathbf{z}_j)$ and returns a noisy gradient $\tilde{\mathbf{g}}_t = \hat{\mathbf{g}}_t + \mathbf{b}_t$ to the optimization oracle Algorithm 1. Throughout the paper, assume:

A1. *The objective function $f : \mathbb{R}^d \to \mathbb{R}$ is bounded from below by $f^\star$ and is L-smooth (L-Lipschitz gradients),* i.e., $\|\nabla f(\mathbf{w}) - \nabla f(\mathbf{w}')\| \leq L\|\mathbf{w} - \mathbf{w}'\|$, *for all $\mathbf{w}, \mathbf{w}' \in \mathcal{W}$.*

A2. *The gradient of ℓ and its noisy approximation are bounded: For all $\mathbf{w} \in \mathcal{W}$, $\mathbf{z} \in \mathcal{Z}$ $\|\nabla\ell(\mathbf{w}, z)\|_1 \leq G_1$ and for all $t \in [T]$, $\|\tilde{\mathbf{g}}_t\|_2 \leq G$.*

High-probability bound: We first show that the noisy gradient $\tilde{\mathbf{g}}_t$ approximates the population gradient $\mathbf{g}_t$ with high probability. A general approach for analyzing such estimation error $|\tilde{\mathbf{g}}_t - \mathbf{g}_t|$ is the Hoeffding's bound. Indeed, given training set $S \in \mathcal{Z}^n$, where $\mathcal{Z} \subset \mathbb{R}$, and a fixed $\mathbf{w}_0$ chosen to be independent of the dataset S, denote the empirical gradient $\hat{\mathbf{g}}_0 = \mathbb{E}_{z \in S}\nabla\ell(\mathbf{w}_0, z)$ and population gradient $\mathbf{g}_0 = \mathbb{E}_{z \sim \mathcal{P}}[\nabla l(\mathbf{w}_0, z)]$ then, Hoeffding's bound implies for coordinate $i \in [d]$ and $\mu > 0$:

$$P\{|\hat{\mathbf{g}}_0^i - \mathbf{g}_0^i| \geq \mu\} \leq 2\exp\left(\frac{-2n\mu^2}{4G_\infty^2}\right) , \tag{2}$$

where G_∞ is the maximal value of the ℓ_∞-norm of the gradient $\mathbf{g}_0$. Generally, if $\mathbf{w}_1$ is updated using the gradient computed on training set S, *i.e.,* $\mathbf{w}_1 = \mathbf{w}_0 - \eta\hat{\mathbf{g}}_0$, concentration inequality (2) *will not* hold for $\hat{\mathbf{g}}_1 = \mathbb{E}_{z\in S}\nabla_i \ell(\mathbf{w}_1, z)$, because $\mathbf{w}_1$ is no longer independent of S. For any differentially private algorithm, Lemma 1 provides the following high probability concentration bound:

Lemma 1. *Let $\mathcal{A}$ be an (ϵ, δ)-differentially private gradient descent algorithm with access to training set S of size n. Let $\mathbf{w}_t = \mathcal{A}(S)$ be the parameter generated at iteration $t \in [T]$ and $\hat{\mathbf{g}}_t$ the empirical gradient on S. For any $\sigma > 0$, $\beta > 0$, if the privacy cost of $\mathcal{A}$ satisfies $\epsilon \leq \sigma/13$, $\delta \leq \sigma\beta/(26\ln(26/\sigma))$, and sample size $n \geq 2\ln(8/\delta)/\epsilon^2$, we then have*

$$\mathbb{P}\left\{|\hat{\mathbf{g}}_t^i - \mathbf{g}_t^i| \geq \sigma\right\} \leq \beta \quad \textit{for every } i \in [d] \textit{ and every } t \in [T]\ .$$

Lemma 1 is an instance of Theorem 8 from [8] and illustrates that, if the privacy cost ϵ is bounded by the estimation error, the differential privacy mechanism enables the reused training samples set to maintain statistical guarantees as if they were fresh samples. Then, we establish in Lemma 2, that SAGD with DPG-LAP is a differentially private algorithm with the following privacy cost:

Lemma 2. SAGD *with* DPG-LAP *(Alg. 1) is* $(\frac{\sqrt{T\ln(1/\delta)}G_1}{n\sigma}, \delta)$*-differentially private.*

In order to achieve a gradient concentration bound for SAGD with DPG-LAP as described in Lemma 1, we set $\sqrt{T\ln(1/\delta)}G_1/(n\sigma) \leq \sigma/13$, $\delta \leq \sigma\beta/(26\ln(26/\sigma))$, and sample size $n \geq 2\ln(8/\delta)/\epsilon^2$. Then, the following result shows that across all iterations, gradients produced by SAGD with DPG-LAP maintain high probability concentration bounds.

Theorem 1. *Given $\sigma > 0$, let $\tilde{\mathbf{g}}_1, ..., \tilde{\mathbf{g}}_T$ be gradients computed by* DPG-LAP *in* SAGD*. Set the number of iterations $2n\sigma^2/G_1^2 \leq T \leq n^2\sigma^4/(169\ln(1/(\sigma\beta))G_1^2)$, then for $t \in [T]$, $\beta > 0$, $\mu > 0$:*

$$\mathbb{P}\left\{\|\tilde{\mathbf{g}}_t - \mathbf{g}_t\| \geq \sqrt{d}\sigma(1+\mu)\right\} \leq d\beta + d\exp(-\mu)\ .$$

Note that given the concentration error bound of $\sqrt{d}\sigma(1+\mu)$, Theorem 1 indicates that a higher noise level σ, implying a better privacy guarantee and a larger number of iterations T,would meanwhile incur a larger concentration error. Thus, there is a trade-off between noise and accuracy illustrated by the positive numbers β and μ. A larger μ brings a larger concentration error but a smaller probability. A larger β implies a larger upper bound on T, yet also a larger probability bound. Note that although the probability $d\beta + d\exp(-\mu)$ has a dependence on dimension d, we can choose appropriate β and μ to make the probability arbitrarily small when analyzing the convergence to a stationary point.

Non-asymptotic convergence rate: We derive the optimal values of σ and T to improve the trade-off between the statistical rate and the optimization rate and we obtain a novel finite-time bound in Theorem 2. Denote $\rho_{n,d} \triangleq \mathcal{O}(\ln n + \ln d)$, we prove that SAGD with DPG-LAP converges to a population stationary point with high probability at the following rate:

Theorem 2. *Given training set S of size n, for $\nu > 0$, if $\eta_t = \eta$ with $\eta \leq \nu/(2L)$, $\sigma = 1/n^{1/3}$, iteration number $T = n^{2/3}/\left(169G_1^2(\ln d + 7\ln n/3)\right)$, $\mu = \ln(1/\beta)$ and $\beta = 1/(dn^{5/3})$, then* SAGD *with* DPG-LAP *algorithm yields:*

$$\min_{1\leq t\leq T} \|\nabla f(\mathbf{w}_t)\|^2 \leq \mathcal{O}\left(\frac{\rho_{n,d}\left(f(\mathbf{w}_1) - f^\star\right)}{n^{2/3}}\right) + \mathcal{O}\left(\frac{d\rho_{n,d}^2}{n^{2/3}}\right),$$

with probability at least $1 - \mathcal{O}\left(1/(\rho_{n,d}n)\right)$.

Theorem 2 shows that, given n samples, SAGD converges to a stationary point at a rate of $\mathcal{O}(1/n^{2/3})$ where we use the ℓ_2 norm of the gradient of the objective function as a convergence criterion. Particularly, the first term of the bound corresponds to the optimization error $\mathcal{O}(1/T)$ with $T = \mathcal{O}(n^{2/3})$, while the second is the statistical error depending on available sample size n and dimension d. The current optimization analyses [36, 34, 39, 5] show that adaptive gradient descent algorithms converge to the stationary point of the objective function with a rate of $\mathcal{O}(1/\sqrt{T})$ with T stochastic gradient computations. Given n samples, their analyses yield a rate of $\mathcal{O}(1/\sqrt{n})$. Thus, the SAGD achieves a sharper bound compared to the previous analyses.

3.2 SAGD with DPG-Sparse

In this section, we consider the SAGD with an advanced version of DPG named DPG-Sparse motivated by the sparse vector technique [7] aiming to provide a sharper result on the privacy cost ϵ and δ. Lemma 2 shows that the privacy cost of SAGD with DPG-Lap scales with $\mathcal{O}(\sqrt{T})$. In order to guarantee the generalization of SAGD as stated in Theorem 1, we need to control the privacy cost below a certain threshold *i.e.,* $\sqrt{T\ln(1/\delta)}G_1/(n\sigma) \leq \sigma/13$. However, it limits the iteration number T of SAGD, leading to a compromised optimization term in Theorem 2. In order to relax the upper bound on T, we propose the SAGD with DPG-Sparse in Algorithm 2. Given n samples, Algorithm 2 splits the dataset evenly into two parts S_1 and S_2. At each iteration t, Algorithm 2 computes gradients on both datasets: $\hat{\mathbf{g}}_{S_1,t} = \frac{1}{|S_1|}\sum_{\mathbf{z}_j \in S_1} \nabla\ell(\mathbf{w}_t, \mathbf{z}_j)$ and $\hat{\mathbf{g}}_{S_2,t} = \frac{1}{|S_2|}\sum_{\mathbf{z}_j \in S_2} \nabla\ell(\mathbf{w}_t, \mathbf{z}_j)$. It then validates $\hat{\mathbf{g}}_{S_1,t}$ with $\hat{\mathbf{g}}_{S_2,t}$, *i.e.,* if the norm of their difference is greater than a random threshold $\tau - \gamma$, it returns $\tilde{\mathbf{g}}_t = \hat{\mathbf{g}}_{S_1,t} + \mathbf{b}_t$, otherwise $\tilde{\mathbf{g}}_t = \hat{\mathbf{g}}_{S_2,t}$.

Algorithm 2 SAGD with DPG-Sparse

```
Input: Dataset $S$, certain loss $\ell(\cdot)$, initial point $\mathbf{w}_0$.
Set noise level $\sigma$, iteration number $T$, and stepsize $\eta_t$.
Split $S$ randomly into $S_1$ and $S_2$.
for $t = 0, ..., T-1$ do
    DPG-Sparse: Compute full batch gradient on $S_1$ and $S_2$:
        $\hat{\mathbf{g}}_{S_1,t} = \frac{1}{|S_1|}\sum_{\mathbf{z}_j \in S_1} \nabla\ell(\mathbf{w}_t, \mathbf{z}_j)$,    $\hat{\mathbf{g}}_{S_2,t} = \frac{1}{|S_2|}\sum_{\mathbf{z}_j \in S_2} \nabla\ell(\mathbf{w}_t, \mathbf{z}_j)$.
    Sample $\gamma \sim \text{Lap}(2\sigma)$, $\tau \sim \text{Lap}(4\sigma)$.
    if $\|\hat{\mathbf{g}}_{S_1,t} - \hat{\mathbf{g}}_{S_2,t}\| + \gamma > \tau$ then
        $\tilde{\mathbf{g}}_t = \hat{\mathbf{g}}_{S_1,t} + \mathbf{b}_t$, where $\mathbf{b}_t^i$ is drawn i.i.d from Lap($\sigma$), for all $i \in [d]$.
    else
        $\tilde{\mathbf{g}}_t = \hat{\mathbf{g}}_{S_2,t}$
    end if
    $\mathbf{m}_t = \tilde{\mathbf{g}}_t$ and $\mathbf{v}_t = (1-\beta_2)\sum_{i=1}^{t} \beta_2^{t-i}\tilde{\mathbf{g}}_i^2$.
    $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \mathbf{m}_t/(\sqrt{\mathbf{v}_t} + \nu)$.
end for
Return: $\tilde{\mathbf{g}}_t$.
```

Following Thresholdout, Zhou et al. [38] propose a stable gradient descent algorithm which uses a similar framework as DPG-Sparse to compute an estimated gradient by validating coordinates of $\hat{\mathbf{g}}_{S_1,t}$ and $\hat{\mathbf{g}}_{S_2,t}$. However, their method is computationally expensive in high-dimensional settings such as deep neural networks. Ours are particularly suited for those models, as observed in Section 4.

High-probability bound: To analyze the privacy cost of DPG-Sparse, let C_s be the number of times the validation fails, *i.e.,* $\|\hat{\mathbf{g}}_{S_1,t} - \hat{\mathbf{g}}_{S_2,t}\| + \gamma > \tau$ is true, over T iterations in SAGD. The following Lemma establishes the privacy cost of the SAGD with DPG-Sparse algorithm.

Lemma 3. SAGD *with* DPG-Sparse *(Alg. 2) is* $(\frac{\sqrt{C_s \ln(2/\delta)}2G_1}{n\sigma}, \delta)$*-differentially private.*

Lemma 3 shows that the privacy cost of SAGD with DPG-Sparse scales with $\mathcal{O}(\sqrt{C_s})$ where $C_s \leq T$. In other words, DPG-Sparse procedure improves the privacy cost of the SAGD algorithm. Indeed, in order to achieve the generalization guarantee of SAGD with DPG-Sparse, stated in Lemma 1 and by considering the result of Lemma 3, we only need to set $\sqrt{C_s \ln(1/\delta)}G_1/(n\sigma) \leq \sigma/13$, which potentially improves the upper bound on T. We derive the generalization guarantee of $\tilde{\mathbf{g}}_t$ generated by the SAGD with DPG-Sparse algorithm in the following result:

Theorem 3. *Given* $\sigma > 0$*, let* $\tilde{\mathbf{g}}_1, ..., \tilde{\mathbf{g}}_T$ *be the gradients computed by* DPG-Sparse *in* SAGD*. With a budget* $n\sigma^2/(2G_1^2) \leq C_s \leq n^2\sigma^4/(676\ln(1/(\sigma\beta))G_1^2)$*, then for* $t \in [T], \beta > 0$*,* $\mu > 0$*:*

$$\mathbb{P}\left\{\|\tilde{\mathbf{g}}_t - \mathbf{g}_t\| \geq \sqrt{d}\sigma(1+\mu)\right\} \leq d\beta + d\exp(-\mu) \ .$$

In the worst case $C_s = T$, we recover the bound of $T \leq n^2\sigma^4/(676\ln(1/(\sigma\beta))G_1^2)$ of DPG-Lap.

Non-asymptotic convergence rate: The finite-time upper bound on the convergence criterion of interest for the SAGD with DPG-Sparse algorithm (Algorithm 2) is stated as follows:

Theorem 4. *Given training set S of size n, for $\nu > 0$, if $\eta_t = \eta$ which are chosen with $\eta \leq \nu/(2L)$, noise level $\sigma = 1/n^{1/3}$, and iteration number $T = n^{2/3}/\left(676G_1^2(\ln d + \frac{7}{3}\ln n)\right)$, then* SAGD *with* DPG-Sparse *algorithm yields:*

$$\min_{1 \leq t \leq T} \|\nabla f(\mathbf{w}_t)\|^2 \leq \mathcal{O}\left(\frac{\rho_{n,d}\left(f(\mathbf{w}_1) - f^\star\right)}{n^{2/3}}\right) + \mathcal{O}\left(\frac{d\rho_{n,d}^2}{n^{2/3}}\right),$$

with probability at least $1 - \mathcal{O}\left(1/(\rho_{n,d}n)\right)$.

Theorem 4 displays a similar rate of $\mathcal{O}(1/n^{2/3})$ for the SAGD with DGP-Sparse as Theorem 2. A sharper bound can be achieved when the number of validation failures C_s is smaller than T. For example, if $C_s = \mathcal{O}(\sqrt{T})$, the upper bound of T can be improved from $T \leq \mathcal{O}(n^2)$ to $T \leq \mathcal{O}(n^4)$.

3.3 Mini-batch Stable Adaptive Gradient Descent Algorithm

For large-scale learning we derive the mini-batch variant of SAGD in Algorithm 3. The training set S is first partitioned into B batches with m samples for each batch. At each iteration t, Algorithm 3 uses any DPG procedure to compute a differential private gradient $\tilde{\mathbf{g}}_t$ on each batch and updates $\mathbf{w}_t$.

Algorithm 3 Mini-Batch SAGD

```
Input: Dataset $S$, certain loss $\ell(\cdot)$, initial point $\mathbf{w}_0$.
Set noise level $\sigma$, epoch number $T$, batch size $m$, and stepsize $\eta_t$.
Split $S$ into $B = n/m$ batches: $\{s_1, ..., s_B\}$.
for $epoch = 1, ..., T$ do
    for $k = 1, ..., B$ do
        Call DPG-Lap or DPG-Sparse to compute $\tilde{\mathbf{g}}_t$.
        $\mathbf{m}_t = \tilde{\mathbf{g}}_t$ and $\mathbf{v}_t = (1 - \beta_2)\sum_{i=1}^{t} \beta_2^{t-i}\tilde{\mathbf{g}}_i^2$.
        $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \mathbf{m}_t/(\sqrt{\mathbf{v}_t} + \nu)$.
    end for
end for
```

Theorem 5. *Consider the mini-batch* SAGD *with* DPG-Lap. *Given S of size n, with $\nu > 0$, $\eta_t = \eta \leq \nu/(2L)$, noise level $\sigma = 1/n^{1/3}$, and epoch $T = m^{4/3}/\left(n169G_1^2(\ln d + \frac{7}{3}\ln n)\right)$, then:*

$$\min_{t=1,\ldots,T} \|\nabla f(\mathbf{w}_t)\|^2 \leq \mathcal{O}\left(\frac{\rho_{n,d}\left(f(\mathbf{w}_1) - f^\star\right)}{(mn)^{1/3}}\right) + \mathcal{O}\left(\frac{d\rho_{n,d}^2}{(mn)^{1/3}}\right),$$

with probability at least $1 - \mathcal{O}\left(1/(\rho_{n,d}n)\right)$.

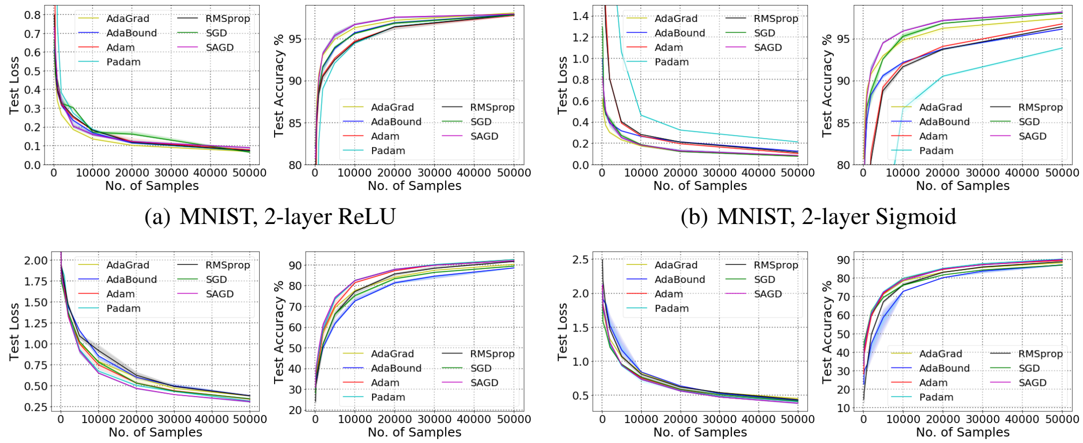
Theorem 5 describes the convergence rate of the mini-batch SAGD algorithm in terms of batch size m and sample size n, *i.e.,* $\mathcal{O}(1/(mn)^{1/3})$. When $m = \sqrt{n}$, mini-batch SAGD achieves the convergence of rate $\mathcal{O}(1/\sqrt{n})$. When $m = n$, *i.e.,* in the full batch setting, Theorem 5 recovers SAGD's convergence rate $\mathcal{O}(1/n^{2/3})$. In terms of computational complexity, the mini-batch SAGD requires $\mathcal{O}(m^{7/3}/n)$ stochastic gradient computations for $\mathcal{O}(m^{4/3}/n)$ passes over m samples, while SAGD requires $\mathcal{O}(n^{5/3})$ stochastic gradient computations. Thus, the mini-batch SAGD has the advantage of decreasing the computation complexity, but displays a slower convergence than SAGD.

4 Numerical Experiments

In this section, we evaluate our proposed mini-batch SAGD algorithm on various deep learning models against popular optimization methods: SGD with momentum [25], Adam [15], Padam [4], AdaGrad [6], RMSprop [31], and Adabound [20]. We consider three tasks: the classification tasks on MNIST [18] and CIFAR-10 [16], and the language modeling task on Penn Treebank [21].

4.1 Environmental Settings

Datasets and Evaluation Metrics: The MNIST dataset has a training set of 60000 examples and a test set of 10000 examples. The CIFAR-10 dataset consists of 50000 training images and 10000 test images. The Penn Treebank dataset contains 929589, 73760, and 82430 tokens for training,



(a) MNIST, 2-layer ReLU

(b) MNIST, 2-layer Sigmoid

(c) CIFAR10, ResNet-18

(d) CIFAR10, VGG-19

Figure 1: **Top row**: Test loss and accuracy of (a) ReLU neural network and (b) Sigmoid neural network on MNIST. The X-axis is the number of train samples, and the Y-axis is the loss/accuracy. In both cases, SAGD obtains the best test accuracy among all the methods. **Bottom row**: Test loss and accuracy of ResNet-18 andVGG-19 on CIFAR10. SAGD achieves the lowest test loss. For VGG-19, SAGD achieves the best test accuracy among all the methods.

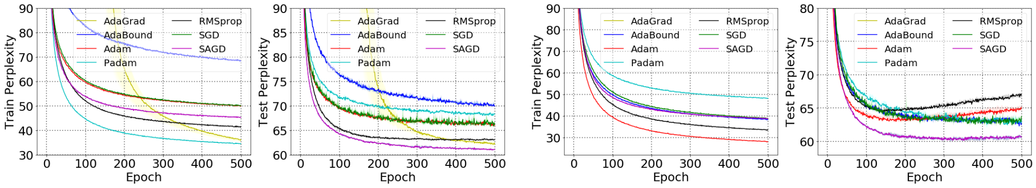
validation, and test, respectively. To better understand the generalization ability of each optimization algorithm with an increasing training sample size n, for each task, we construct multiple training sets of different size by sampling from the original training set. For MNIST, training sets of size $n \in \{50, 100, 200, 500, 10^3, 2.10^3, 5.10^3, 10^4, 2.10^4, 5.10^4\}$ are constructed. For CIFAR10, training sets of size $n \in \{200, 500, 10^3, 2.10^3, 5.10^3, 10^4, 2.10^4, 3.10^4, 5.10^4\}$ are constructed. For each n, we train the model and report the loss and accuracy on the test set. For Penn Treebank, all training samples are used to train the model and we report the training perplexity and the test perplexity across epochs. Cross-entropy is used as the loss function throughout experiments. The mini-batch size is set to be 128 for CIFAR10 and MNIST, 20 for Penn Treebank. We repeat each experiment 5 times and report the mean and standard deviation of the results.

Hyper-parameter setting: Optimization hyper-parameters affect the quality of solutions. Particularly, Wilson et al. [35] highlight that the initial stepsize and the scheme of decaying stepsizes have a considerable impact on the performance. We follow the logarithmically-spaced grid method in Wilson et al. [35] to tune the stepsize. If the parameter performs best at an extreme end of the grid, a new grid will be tried until the best parameter lies in the middle of the grid. Once the interval of the best stepsize is located, we change to the linear-spaced grid to further search for the optimal one. We specify the strategy of decaying stepsizes in the subsections of each task. For each experiment, we set $\sigma^2 = 1/n^{2/3}$, where n is the size of the training set, as stated in Theorem 5. Parameters ν, β_2, and T follow the default settings as adaptive algorithms such as RMSprop.

4.2 Numerical results

Feedforward Neural Network. For image classification on MNIST, we focus on two 2-layer fully connected neural networks with either ReLU or Sigmoid activation functions. We run 100 epochs and decay the learning rate by 0.5 every 30 epochs. Figure 1 presents the loss and accuracy on the test set given different training set sizes. Since all algorithms attain the 100% training accuracy, the performance on the training set is omitted. Figure 1 (a) shows that, for ReLU neural network, SAGD performs slightly better than the other algorithms in terms of test accuracy. When $n = 50000$, SAGD gets a test accuracy of $98.38 \pm 0.13\%$. Figure 1 (b) presents the results on Sigmoid neural network where SAGD achieves the best test accuracy among all the algorithms. When $n = 50000$, SAGD reaches the highest test accuracy of $98.14 \pm 0.11\%$, outperforming other adaptive algorithms.

Convolutional Neural Network. We use ResNet-18 [13] and VGG-19 [30] for the CIFAR-10 image classification task. We run 100 epochs and decay the learning rate by 0.1 every 30 epochs. The results are presented in Figure 1. Figure 1 (c) shows that SAGD has higher test accuracy than the other algorithms when the sample size is small *i.e.,* $n \leq 20000$. When $n = 50000$, SAGD



(a) Penn Treebank, 2-Layer LSTM

(b) Penn Treebank, 3-Layer LSTM

Figure 2: Train and test perplexity of 2-layer LSTM and 3-layer LSTM. Although adaptive methods such as AdGrad, Padam, Adam, and RMSprop achieves better training performance than SAGD, SAGD performs the best in terms of the test perplexity among all the methods.

achieves nearly the same test accuracy, $92.48 \pm 0.09\%$, as Adam, Padam, and RMSprop. Non-adaptive algorithm SGD performs better than the other algorithms in terms of test loss. Figure 1 (d) reports the results on VGG-19. Although SAGD has a higher test loss than the other algorithms, it achieves the best test accuracy, especially when n is small. Non-adaptive algorithm SGD performs better than the other adaptive gradient algorithms regarding the test accuracy. When $n = 50000$, SGD has the best test accuracy $91.36 \pm 0.04\%$. SAGD achieves accuracy $91.26 \pm 0.05\%$.

Recurrent Neural Network. Finally, an experiment on Penn Treebank is conducted for the language modeling task with 2-layer Long Short-Term Memory (LSTM) [22] network and 3-layer LSTM. We train them for a fixed budget of 500 epochs and omit the learning-rate decay. Perplexity is used as the metric to evaluate the performance and learning curves are plotted in Figure 2. Figure 2 (a) shows that for the 2-layer LSTM, AdaGrad, Padam, RMSprop and Adam achieve a lower training perplexity than SAGD. However, SAGD performs the best in terms of the test perplexity. Specifically, SAGD achieves 61.02 ± 0.08 test perplexity. In particular, we observe that after 200 epochs, the test perplexity of AdaGrad and Adam starts increasing, but the training perplexity continues decreasing (over-fitting occurs). Figure 2 (b) reports the results for the 3-layer LSTM. We can see that the perplexity of AdaGrad, Padam, Adam, and RMSprop start increasing significantly after 150 epochs (*over-fitting*) while the perplexity of SAGD keeps decreasing. SAGD, SGD and AdaBounds perform better than AdaGrad, Padam, Adam, and RMSprop in terms of over-fitting. Table 1 shows the best test perplexity of 2-layer LSTM and 3-layer LSTM for all the algorithms. We can observe that the SAGD achieves the best test perplexity 59.43 ± 0.24 among all the algorithms.

Table 1: Test Perplexity of LSTMs on Penn Treebank. Bold number indicates the best result.

	RMSprop	Adam	AdaGrad	Padam	AdaBound	SGD	SAGD
2-layer LSTM	62.87 ± 0.05	60.58 ± 0.37	62.20 ± 0.29	62.85 ± 0.16	65.82 ±0.08	65.96 ± 0.23	**61.02 ± 0.08**
3-layer LSTM	63.97 ± 018	63.23 ± 004	66.25 ± 0.31	66.45 ± 0.28	62.33 ±0.07	62.51 ±0.11	**59.43 ± 0.24**

5 Conclusion

In this paper, we focus on the generalization ability of adaptive gradient methods. Concerned with the observation that adaptive gradient methods generalize worse than SGD for over-parameterized neural networks and given the limited theoretical understanding of the generalization of those methods, we propose **S**table **A**daptive **G**radient **D**escent (SAGD) methods, which boost the generalization performance in both theory and practice through a novel use of differential privacy. The proposed algorithms generalize well with provable high-probability convergence bounds of the population gradient. Experimental studies highlight that the proposed algorithms are competitive and often better than baseline algorithms for training deep neural networks and demonstrate the aptitude of our method to avoid over-fitting through a differential privacy mechanism.

6 Broader Impact

We believe that our work stands in the line of several papers towards improving generalization and avoiding over-fitting. Indeed, the basic principle of our method is to fit any given model, in particular deep model, using an intermediate differentially-private mechanisms allowing the model to fit fresh samples while passing over the same batch of n observations. The impact of such work is straightforward and could avoid learning, and thus reproducing at testing phase, the bias existent in the training dataset.

References

[1] R. Bassily, A. Smith, and A. Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 464–473. IEEE, 2014.

[2] O. Bousquet and A. Elisseeff. Stability and generalization. *Journal of machine learning research*, 2(Mar):499–526, 2002.

[3] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(Mar):1069–1109, 2011.

[4] J. Chen and Q. Gu. Closing the generalization gap of adaptive gradient methods in training deep neural networks. *arXiv preprint arXiv:1806.06763*, 2018.

[5] X. Chen, S. Liu, R. Sun, and M. Hong. On the convergence of a class of adam-type algorithms for non-convex optimization. In *International Conference on Learning Representations*, 2019.

[6] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[7] C. Dwork, A. Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[8] C. Dwork, V. Feldman, M. Hardt, T. Pitassi, O. Reingold, and A. Roth. Generalization in adaptive data analysis and holdout reuse. In *Advances in Neural Information Processing Systems*, pages 2350–2358, 2015.

[9] C. Dwork, V. Feldman, M. Hardt, T. Pitassi, O. Reingold, and A. Roth. The reusable holdout: Preserving validity in adaptive data analysis. *Science*, 349(6248):636–638, 2015.

[10] C. Dwork, V. Feldman, M. Hardt, T. Pitassi, O. Reingold, and A. L. Roth. Preserving statistical validity in adaptive data analysis. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 117–126. ACM, 2015.

[11] S. Ghadimi and G. Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

[12] M. Hardt, B. Recht, and Y. Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *International Conference on Machine Learning*, pages 1225–1234, 2016.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[14] N. S. Keskar and R. Socher. Improving generalization performance by switching from adam to sgd. *arXiv preprint arXiv:1712.07628*, 2017.

[15] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *In Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

[16] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[17] I. Kuzborskij and C. Lampert. Data-dependent stability of stochastic gradient descent. In *International Conference on Machine Learning*, pages 2820–2829, 2018.

[18] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[19] J. Li, X. Luo, and M. Qiao. On generalization error bounds of noisy gradient methods for non-convex learning. *arXiv preprint arXiv:1902.00621*, 2019.

[20] L. Luo, Y. Xiong, and Y. Liu. Adaptive gradient methods with dynamic bound of learning rate. In *International Conference on Learning Representations*, 2019.

[21] M. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of english: the penn treebank. *Computational linguistics-Association for Computational Linguistics*, 19(2):313–330, 1993.

[22] S. Merity, N. S. Keskar, and R. Socher. Regularizing and optimizing LSTM language models. In *International Conference on Learning Representations*, 2018.

[23] W. Mou, L. Wang, X. Zhai, and K. Zheng. Generalization bounds of sgld for non-convex learning: Two theoretical viewpoints. In *Conference On Learning Theory*, pages 605–638, 2018.

[24] A. Pensia, V. Jog, and P.-L. Loh. Generalization error bounds for noisy, iterative algorithms. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pages 546–550. IEEE, 2018.

[25] N. Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

[26] M. Raginsky, A. Rakhlin, and M. Telgarsky. Non-convex learning via stochastic gradient langevin dynamics: a nonasymptotic analysis. In *Conference on Learning Theory*, pages 1674–1703, 2017.

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

[28] H. Robbins and S. Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[29] S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[30] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[31] T. Tieleman and G. Hinton. Rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 2012.

[32] D. Wang and J. Xu. Differentially private empirical risk minimization with smooth non-convex loss functions: A non-stationary view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1182–1189, 2019.

[33] D. Wang, M. Ye, and J. Xu. Differentially private empirical risk minimization revisited: Faster and more general. In *Advances in Neural Information Processing Systems*, pages 2722–2731, 2017.

[34] R. Ward, X. Wu, and L. Bottou. Adagrad stepsizes: sharp convergence over nonconvex landscapes. In *International Conference on Machine Learning*, pages 6677–6686, 2019.

[35] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht. The marginal value of adaptive gradient methods in machine learning. In *Advances in Neural Information Processing Systems*, pages 4148–4158, 2017.

[36] M. Zaheer, S. Reddi, D. Sachan, S. Kale, and S. Kumar. Adaptive methods for nonconvex optimization. In *Advances in Neural Information Processing Systems*, pages 9793–9803, 2018.

[37] D. Zhou, Y. Tang, Z. Yang, Y. Cao, and Q. Gu. On the convergence of adaptive gradient methods for nonconvex optimization. *arXiv preprint arXiv:1808.05671*, 2018.

[38] Y. Zhou, S. Chen, and A. Banerjee. Stable gradient descent. In *UAI*, pages 766–775, 2018.

[39] F. Zou, L. Shen, Z. Jie, W. Zhang, and W. Liu. A sufficient condition for convergences of adam and rmsprop. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11127–11135, 2019.