Unlocking the Power of Differentially Private Zeroth-order Optimization for Fine-tuning LLMs

Ergute Bao[*], *Yangfan Jiang*[†], *Fei Wei*[*], *Xiaokui Xiao*[†], *Zitao Li*[*], *Yaliang Li*[*], *Bolin Ding*[*]
[*]*Alibaba Group,* [†]*National University of Singapore*

Abstract

Differentially private zeroth-order optimization (DPZero in short) has shown promise in fine-tuning large language models (LLMs) while protecting record-level privacy. Compared with classical first-order methods, such as DPSGD, the main difference is that DPZero replaces the exact first-order gradients that are computed via back-propagation with its random zeroth-order approximations that are computed via querying the model's losses. However, DPZero still lags in the resulting model utility compared to existing methods, indicating that further work is needed to fully realize its potential.

In this paper, we make a solid step towards designing a better differentially private algorithm for fine-tuning LLMs based on zeroth-order optimization. Our design is centered around the major performance issue of differentially private optimization for large models caused by artificial clipping, which creates biases in the model updates. Using our method called *DP-AggZO*, we theoretically prove that this issue can be mitigated, leading to an improved convergence rate over the prior DPZero methods and better model utility under the same privacy constraints. We back up our theory with extensive experiments, validating the performance improvement of *DP-AggZO*. Surprisingly, our *DP-AggZO* even outperforms the state-of-the-art method DP-AdamW significantly on some benchmark settings.

1 Introduction

Despite the success of large language models (LLMs) in diverse tasks (e.g., see [7,15,26,29,43,46,47,75,100]), there are growing concerns regarding the security of LLMs, particularly in terms of preventing potential misuse [9,103,107], detecting machine-generated texts [48, 55, 80], and mitigating demographical biases in generated content [95]. In addition, since LLMs are built upon large-scale neural networks, they also inherit the security and privacy risks inherent to large models, including vulnerabilities in robustness [112], data/model poisoning [20,40,86,87], backdoor injections [8,11,67,74,117], data/model extraction [22, 25, 53, 76, 99], membership inference attacks [21,62,73,88,91,109], and data memorization [23,90].

In this work, we address the data privacy aspect of LLMs, which has received increasing attention from both the research community and governmental agencies (e.g., see reports on data protection for LLMs [56] and ChatGPT [13]). Notably, LLMs are known to be vulnerable to membership inference attacks [57] and data extraction attacks (both in the lab [24] and in production [72]), leaking personally identifiable information [23,66] from the training dataset and violating privacy regulations GDPR [49] and CCPA [45].

Our goal is to provide an efficient and effective algorithm for *fine-tuning LLMs* while respecting *record-level privacy*, preventing an adversary from inferring the membership or reconstructing a data point from the fine-tuning dataset. To achieve this, we adopt differential privacy (DP) [38], a well-established privacy framework that provides provable protection against membership inference and data reconstruction attacks [24,34,72,88]. DP has been applied to safeguarding data privacy in deep learning [1,32,102,105,106] and, more recently, to both training LLMs from scratch and fine-tuning LLMs on specific tasks [6,59,97,110].

The general idea of DP is to obfuscate the contribution of an individual record in the result released to the adversary, via random noise injection (see Section 2.1 for details). In particular, to achieve a designated level of privacy, the scale of the injected noise must be proportional to the maximum contribution of an individual record to the algorithm's output [38]. In this way, the adversary cannot infer with high confidence the presence or absence of any particular record in the input dataset, thereby ensuring record-level privacy. However, implementing this idea in the training or fine-tuning process of neural networks is not straightforward, as it is challenging to analytically quantify the maximum contribution of a record to the output of the training/fine-tuning process — i.e., the final model parameters — due to the complex nature of general neural networks. On the other hand, an inappropriate selection for the scale of injected noise leads to unsatisfactory

privacy-utility trade-offs: if the noise is much larger than the actual contribution made by an individual record, the model's utility is sacrificed due to the overwhelmingly large noise; otherwise, if the scale of the noise is much smaller than the individual contribution, privacy is at risk.

DPSGD [1] resolves the above challenge as follows. First, instead of quantifying the overall contribution of an individual record to the final model parameters, *DPSGD* quantifies its contribution to the gradient sum obtained at each iteration, i.e., the individual gradient, which is used for model updates. However, due to the complex nature of the loss functions and the models, this process is still not straightforward. This is where artificial clipping comes into play. Specifically, each individual gradient corresponding to a training record is clipped to a predefined range: if the $\mathcal{L}_2$ norm of the gradient exceeds a predefined constant c, the gradient is rescaled to have a norm of c; otherwise, it remains unchanged. After clipping each individual gradient, independent random Gaussian noise is injected into each dimension of the sum of the clipped gradients and the scale of the noise (namely, the standard deviation of the Gaussian noise) is proportional to the clipping threshold c to ensure DP, preventing adversaries from inferring the training data [111]. Finally, the model is updated according to the estimated gradient average from the noisy gradient sum, and this process is repeated for T iterations. Note that the privacy reasoning of *DPSGD* is based on the noisy gradient sum in each model update rather than the final model parameters. Assuming that the adversary observes such statistics and knows the algorithm as public information, then he could compute the model parameters by himself without incurring additional privacy costs. Indeed, this analysis assumes nothing about the model architecture and the loss function and has been widely used for deep learning with DP (we refer interested readers to, e.g., [28, 41], for more delicate analyses tailored to specific model architectures and loss functions).

Akin to its non-DP version *Mini-batch SGD*, fine-tuning models, especially language models, with *DPSGD* consumes large computational resources (compared with querying/inference with the models), due to the backpropagation required to compute individual gradients. In particular, even for medium- to small-sized open-source models, such as OPT-6.7B [115], the gradient computation exceeds the memory capabilities of high-end GPUs such as the H100 with 80GB of RAM. This limitation also imposes additional challenges for preserving data privacy, explained as follows. Imagine that a small research team would like to release a model fine-tuned on their sensitive data under differential privacy for public usage. However, due to the lack of computational resources, the fine-tuning process has to be outsourced to untrusted cloud service providers, during which the sensitive data could be exposed. Such constraints create significant barriers for researchers, preventing them from safely fine-tuning and sharing models based on sensitive data, limiting broader access to their work. Recent advances in memory-efficient DP fine-tuning techniques, such as *DP-LoRA* and *DP-Prefix Tuning* [59, 110], have aimed to address these limitations.

Memory-efficient Zeroth-order optimization (*MeZO*), originating from control theory [39, 92], has recently demonstrated promising performance in fine-tuning large language models [68] while incurring only a small memory overhead compared to model querying. The overall idea of MeZO [68] is to replace the exact *first-order* gradients that are costly to compute in SGD [85] with its random zeroth-order approximations that can be computed by querying the model's losses in some neighborhoods. Given that evaluating the losses is often less computationally intensive than computing the gradient, zeroth-order optimization imposes fewer constraints on the computation resources. This improvement is particularly critical for fine-tuning large language models, where the model itself consumes a significant amount of GPU memory, not to mention the computation of its gradients (the latter sometimes requires around $\times 12$ more memory than querying the model, as reported in [68]).

Perhaps due to this reason, since the release of MeZO [68], it has been quickly adopted by the differential privacy (DP) literature–several differentially private zeroth-order optimization methods have been proposed. The original DP-version of the Zeroth-order algorithm was named *DPZero* in the workshop version of [113]. Variants of *DPZero* were later studied in [63, 96]. Inherited from *MeZO*, *DPZero* methods requires far less memory compared with *DPSGD* when fine-tuning large models. While the initial results of *DPZero* show promise, there remain a few noticeable utility gaps between *DPZero* and traditional first-order methods in certain benchmark settings, indicating that further work is needed to unlock their full potential.

In summary, memory consumption and data privacy are critical considerations when fine-tuning large language models. On the one hand, *DPSGD* demonstrates promising performance (i.e., measured as the model's test accuracy/utility under specific DP constraints) while incurring substantial memory overhead compared with querying with the models, creating barriers for researchers and practitioners with limited access to high-end hardware. On the other hand, while *DPZero* methods demonstrate a significant reduction in terms of memory requirements, their performance is still considerably lower than the classic first-order state-of-the-art *DPSGD*. The question then becomes: can we design more powerful *DPZero* methods that achieve higher performance (i.e., better privacy-utility trade-offs) without incurring significant memory overhead? We give a positive answer in this paper.

1.1 Our Contribution

In this paper, we take a solid step towards designing better memory-efficient differentially private methods for fine-tuning large language models based on the zeroth-order optimization (i.e., ZO). The core design of our approach, *DP-*

AggZO, is based on reducing the error introduced by artificial clipping in differential privacy for zeroth-order optimization. Toward that end, in Section 4, we first establish a quantitative relationship between this clipping error and the model's convergence rate (which is also of independent interest), motivating our design. However, as is also observed in prior DP algorithms, reducing the clipping error often increases the error from additive DP noises, which does not guarantee utility increase under same privacy constraints. Our *DP-AggZO*, detailed in Section 5, resolves this challenge via algorithmic level modifications to the original *DPZero*. Through rigorous analysis, we prove that our *DP-AggZO* theoretically exhibits a faster convergence rate than existing *DPZero* methods under the same privacy constraints without requiring more GPU memory.

We also conduct comprehensive experiments to validate the performance of *DP-AggZO* in Section 6. Notably, when fine-tuning the OPT-6.7B model, *DPSGD* fails to run on a 96-GB GPU due to its large memory consumption, while our *DP-AggZO* excels, offering a significant improvement over the recent *DPZero* baseline in terms of privacy-utility trade-offs. On the OPT-1.3B model, *DP-AggZO* matches, and in some cases, surpasses the state-of-the-art *DP-AdamW* (i.e., *DPSGD* with the Adam optimizer); and on RoBERTa (355M), *DP-AggZO* outperforms the state-of-the-art *DP-AdamW* on all five benchmark datasets. Overall, *DP-AggZO* consistently outperforms existing methods. Our work sets new benchmarks for fine-tuning large language models under differential privacy without creating barriers on hardware, enabling a wider audience (including those from under-resourced environments) to benefit from privacy-preserving technology.

2 Preliminaries

Notations. We use $\mathcal{D}$ to denote the input dataset (i.e., the dataset for fine-tuning an LLM) and use $n = |\mathcal{D}|$ to denote the number of records in $\mathcal{D}$. We use $\mathcal{B}$ to denote a batch (i.e., a subset of $\mathcal{D}$). Given a differentiable loss function $\mathcal{L} : \mathbb{R}^d \times \mathcal{X} \to \mathbb{R}$, we denote $\mathcal{L}(\boldsymbol{\theta}; x)$ and $\nabla\mathcal{L}(\boldsymbol{\theta}; x)$ as the loss and the first-order gradient computed on $\boldsymbol{\theta}$ for record x, respectively. We use $\|\mathbf{v}\|$ to denote the $\mathcal{L}_2$ norm of vector $\mathbf{v}$.

2.1 Differential Privacy

We say that two datasets $\mathcal{D}_1$ and $\mathcal{D}_2$ are neighboring if one can be obtained by adding or removing a record in the other, denoted as $\mathcal{D}_1 \sim \mathcal{D}_2$. Differential privacy (DP) [38] enforces an upper bound on the distance between the output distributions of a mechanism M on neighboring datasets.

Definition 1 (Differential Privacy [38]). *A randomized mechanism M is (ε, δ)-differentially private if for any neighboring datasets $\mathcal{D}_1$ and $\mathcal{D}_2$ that differ by one record, and any subset O of the output domain of M, we have the following.*

$$\Pr[M(\mathcal{D}_1) \in O] \leq \exp(\varepsilon) \cdot \Pr[M(\mathcal{D}_2) \in O] + \delta. \quad (1)$$

Parameters ε and δ are referred to as the privacy constraints. Smaller values of ε and δ make it more difficult to distinguish the input (either $\mathcal{D}_1$ or its neighbor $\mathcal{D}_2$) from the output of M, implying stronger privacy protections, and vice versa.

An alternative DP framework, Rényi differential privacy (RDP or Rényi-DP) [70], built upon the concept of Rényi divergence [84], is commonly used for analyzing iterative algorithms, which are especially common for fine-tuning.

Definition 2 (RDP [70]). *A randomized mechanism M satisfies (α, ε)-RDP if for any neighboring datasets $\mathcal{D}_1$ and $\mathcal{D}_2$ that differ by one record, we have the following.*

$$\mathrm{D}_\alpha(M(\mathcal{D}_1) \,\|\, M(\mathcal{D}_2)) \leq \varepsilon, \quad (2)$$

where $\mathrm{D}_\alpha(P \,\|\, Q)$ represents the Rényi divergence between probability distributions P and Q.

The canonical approach to turn a function F into its DP version is by noise injection, where the scale of the noise is calibrated to the sensitivity of the function, defined as follows.

Definition 3 ($\mathcal{L}_2$ sensitivity). *The $\mathcal{L}_2$ sensitivity of a function F is defined as the maximum change caused by adding/removing a record in the input.*

$$S(F) = \max_{\mathcal{D}_1 \sim \mathcal{D}_2} \|F(\mathcal{D}_1) - F(\mathcal{D}_2)\|. \quad (3)$$

Lemma 1 (Gaussian noise satisfies RDP [70]). *Injecting Gaussian noise $\mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_d)$ to any d-dimensional function with bounded $\mathcal{L}_2$ sensitivity Δ_2 achieves $\left(\alpha, \frac{\alpha\Delta_2^2}{2\sigma^2}\right)$-RDP.*

Both RDP and (ε, δ)-DP are preserved under post-processing, Further details of DP are in the technical report version [12].

2.2 DPSGD for Fine-tuning

Given a differentiable loss function $\mathcal{L}$, we aim to find the optimal solution (i.e., model parameters) that minimizes the overall loss over input dataset $\mathcal{D}$, i.e., finding the solution to $\arg\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \sum_{x \in \mathcal{D}} \mathcal{L}(\boldsymbol{\theta}; x)$.

Mini-batch SGD (non-DP). The standard non-DP approach, *mini-batch SGD*, aims to solve the above optimization problem by iteratively refining its solution. In particular, at iteration t, a random subset of $\mathcal{D}$ is sampled to form a mini batch $\mathcal{B}_t$ (say, using the Poisson sampling with sampling rate q). Next, the gradient sum $\sum_{x \in \mathcal{B}_t} \nabla\mathcal{L}(\boldsymbol{\theta}_t; x)$ is computed over records in $\mathcal{B}_t$. With learning rate η, the current solution $\boldsymbol{\theta}_t$ is updated to $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \frac{\eta}{|\mathcal{B}_t|} \cdot \sum_{x \in \mathcal{B}_t} \nabla\mathcal{L}(\boldsymbol{\theta}_t; x)$. In the context of fine-tuning a large language model (LLM), we can denote θ_0,

i.e., the initial solution, as the parameters of the pre-trained model, e.g., RoBERTa (355M) [61].

DPSGD. Roughly speaking, *DPSGD* [1] enforces differential privacy over the computation for the gradient sum in mini-batch SGD. *DPSGD* first enforces a bound on the $\mathcal{L}_2$ norm of individual gradients through *artificial clipping*. In particular, we are given a pre-fixed positive constant c, called the clipping threshold: for each individual record's gradient, if its norm exceeds c, then it is rescaled to have a norm of c; otherwise, it remains unchanged. The clipped gradient is

$$\widehat{\nabla}\mathcal{L}(\boldsymbol{\theta}_t;x) = \frac{\min(c, \|\nabla\mathcal{L}(\boldsymbol{\theta}_t;x)\|)}{\|\nabla\mathcal{L}(\boldsymbol{\theta}_t;x)\|}\nabla\mathcal{L}(\boldsymbol{\theta}_t;x). \tag{4}$$

Now that the sensitivity for the *function of gradient sum* is bounded by c, we can enforce DP by perturbing the sum of clipped gradients with a random noise sampled from $\mathcal{N}(\mathbf{0}, \sigma_m^2 c^2 \mathbf{I}_d)$. Here, σ_m is termed as the *noise multiplier* [71], which is computed from the designated DP guarantee, the number of iterations, and the subsampling rate for batches. We note that σ_m is independent of c. The DP gradient sum is

$$G_{\text{dpsgd}}(\boldsymbol{\theta}_t;\mathcal{B}_t) = \left(\sum_{x\in\mathcal{B}_t}\widehat{\nabla}\mathcal{L}(\boldsymbol{\theta}_t;x)\right) + \mathcal{N}(\mathbf{0}, \sigma_m^2 c^2 \mathbf{I}_d). \tag{5}$$

The model is then updated according to G_{dpsgd} and this process is repeated for some T iterations. We defer further details on DP and *DPSGD* to technical report version [12].

In *DPSGD*, the memory consumption of computing the model's gradient via backpropagation is $O(M^2)$, where M stands for the size of the largest layer in the model. This can be prohibitive for large models and cause out-of-memory issues on small GPUs. Parameter-efficient fine-tuning methods such as *DP-LoRA* and *DP-Prefix Tuning* [59, 110] improve the memory and computation costs of *DPSGD* while other methods such as [17] optimize the computation of *DPSGD* on specific model structures. While these methods improve the efficiency of *DPSGD*, they do not lead to better utility of the fine-tuned model under the same privacy constraints.

2.3 Zeroth-order Fine-tuning

Unlike first-order methods, zeroth-order optimization methods (*ZO*, in short) do not require gradient computation, e.g., see [39, 92]. Instead, *ZO* approximates the gradients using differences of the loss function evaluated at points around $\boldsymbol{\theta}_t$, based on Taylor expansion.

MeZO (non-DP). Recent work *MeZO* [68] adapts this idea to fine-tuning LLMs. In particular, for each record x, the gradient $\nabla\mathcal{L}(\boldsymbol{\theta}_t;x)$ is approximated as

$$\frac{\mathcal{L}(\boldsymbol{\theta}_t + \phi\mathbf{z};x) - \mathcal{L}(\boldsymbol{\theta}_t - \phi\mathbf{z};x)}{2\phi}\cdot\mathbf{z} \approx \nabla\mathcal{L}(\boldsymbol{\theta}_t;x). \tag{6}$$

Here ϕ, referred to as the perturbation scale, is some pre-fixed positive constant, and $\mathbf{z}$, referred to as the perturbation vector, is sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I}_d)$.

To obtain the above result, given $\mathbf{z}$ and ϕ, we need to compute the model's loss difference in its neighborhood $\boldsymbol{\theta}_t \pm \phi\mathbf{z}$

$$\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t;x) = \frac{\mathcal{L}(\boldsymbol{\theta}_t + \phi\mathbf{z};x) - \mathcal{L}(\boldsymbol{\theta}_t - \phi\mathbf{z};x)}{2\phi}, \tag{7}$$

which is an approximation to the loss's directional derivative along $\mathbf{z}$. The following proposition (can be proved using Taylor's theorem) roughly speaks for the convergence of ZO.

Proposition 1. *Let $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$, then when $\phi \to 0$, we have $\mathbb{E}[\Delta_{\mathbf{z}}(\theta_t;x)\cdot\mathbf{z}] = \nabla\mathcal{L}(\theta_t;x)$,*

In our later analysis for model's convergence, we adopt the assumption that $\mathbb{E}[\Delta_{\mathbf{z}}(\theta_t;x)\cdot\mathbf{z}] = \nabla\mathcal{L}(\theta_t;x)$, as is the convention in the literature (e.g., see [37, 68, 113, 114]). Finally, one can also sample multiple vectors $\{\mathbf{z}_k\}_{k=1}^K$ independently from $\mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ and use $\frac{1}{K}\sum_{k=1}^K \Delta_{\mathbf{z}_k}(\theta_t;x)\cdot\mathbf{z}_k$ as the estimation for $\nabla\mathcal{L}(\boldsymbol{\theta}_t;x)$. According to [68], this variant of ZO does not lead to performance improvement for non-DP fine-tuning when the overall computation is fixed.

Differentially private ZO. To preserve DP, *DPZero* [96] (referred to as *DPZero* afterwards) proceed as follows. First, given $\mathbf{z}$, for each record x in the sampled batch $\mathcal{B}_t$, we clip the scale of $\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t;x)$ with some threshold c, obtaining

$$\widehat{\Delta}_{\mathbf{z}}(\boldsymbol{\theta}_t;x) = \frac{\min(c, |\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t;x)|)}{|\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t;x)|}\cdot\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t;x). \tag{8}$$

With the pre-fixed noise multiplier σ_m, a one-dimensional random Gaussian noise $u \sim \mathcal{N}(0, \sigma_m^2 c^2)$ is injected to the sum $\sum_{x\in\mathcal{B}_t}\widehat{\Delta}_{\mathbf{z}}(\boldsymbol{\theta}_t;x)$ to preserve the designated level of privacy. Note that the vector $\mathbf{z}$ is seen as public information and does not require privacy protection. The model is updated to

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta\frac{\left(\sum_{x\in\mathcal{B}_t}\widehat{\Delta}_{\mathbf{z}}(\boldsymbol{\theta}_t;x)\right) + u}{b}\cdot\mathbf{z}, \tag{9}$$

where b is the targeted batch size. When using Poisson sampling (see e.g., [30, 71]), the targeted batch size equals to qn (where q stands for the sampling rate), which also reveals information of $\mathcal{D}$ and hence, needs privacy protections. We often replace b with $q\tilde{n}$, where $\tilde{n}$ stands for the DP estimate for n. The result of $\tilde{n}$ can be reused without incurring additional DP costs. The overall privacy cost (i.e., the privacy parameters) for running *DPZero* for T iterations is the sum of costs for releasing $\tilde{n}$ for once and releasing the DP estimate for $\sum_{x\in\mathcal{B}_t}\widehat{\Delta}_{\mathbf{z}}(\boldsymbol{\theta}_t;x)$ in all T iterations (see the full version [12] for more details).

Memory reduction. Compared with the first-order methods *mini-batch SGD* and *DPSGD*, the memory reduction of zeroth-order methods is significant, because only forward passes are involved and there is no computation of gradients using backpropagation (see [68] for more details). To get $\theta_t \pm \phi\mathbf{z}$, there is also no need to store the actual outcome of $\mathbf{z} \in \mathbb{R}^d$. Instead, one could just store the random seed for generating

$\mathbf{z}$ and then iteratively sample particular dimensions of $\mathbf{z}$ for each layer of the model and then apply the change in an in-line manner. This process incurs a memory overhead of $O(M)$ only (where M stands for the size of the largest layer). Without DP, *MeZO* [68] reports up to $\times 12$ memory reduction (measured as GB) compared with *SGD*, and *DPZero* [113] reports around $\times 8$ reduction of GPU memory compared with *DPSGD* on the benchmarking model RoBERTa 355M [61].

Implications on data privacy. The memory reduction of *DPZero* also enables data curators to fine-tune larger models on their sensitive data using (local) machines of limited resources, instead of resorting to outsourced computation such as cloud services (with larger GPUs), which, in turn, may bring more challenges to safeguarding data privacy (e.g., see [81, 94]).

3 Problem Definition

Problem. We study the problem of fine-tuning large language models (LLMs) with differential privacy. In particular, given a pre-trained model with initial parameters $\boldsymbol{\theta}_0$, dataset $\mathcal{D}$, and loss function $\mathcal{L}$, we want to fine-tune the model to minimize its loss computed over $\mathcal{D}$, while respecting record-level differential privacy. In addition, we would like to restrict the optimization algorithm to querying the model's losses only, rather than computing the exact gradients via backpropagation, following prior work on zeroth-order optimization for reducing the memory usage [63, 68, 96, 113].

Motivation. In the previous work [63, 96, 113], the differentially private zeroth-order optimization method demonstrates promising performance when fine-tuning LLMs. Despite that, when compared with the state-of-the-art first-order method *DPSGD*, the utility gap is notable. This brings us to this question: *can we further improve the performance of zeroth-order optimization for fine-tuning LLMs with differential privacy?*

In this work, we answer the above question positively. Specifically, we proposed an improved algorithm for ZO fine-tuning with DP. Overall, our method achieves a faster convergence rate (in theory) and demonstrates empirical improvement over prior methods (in practice) under the same privacy constraints.

4 Motivation and Idea

In what follows, we first present a new convergence analysis for *DPZero* [113]. Guided by the analysis, we introduce the idea to our solution of differentially private zeroth-order fine-tuning (detailed in Section 5).

4.1 Revisiting the Convergence of DPZero

Empirical observation. Existing analysis of *DPZero* often assumes that the clipping error can be ignored (e.g., [113]). In reality, however, this assumption breaks down under typical settings. In particular, prior work [96, 113] recommended clipping thresholds such as $c = 100$ and 200. However, Figure 1 shows that the distribution of $|\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)|$ is heavily tailed. As a result, a substantial fraction of samples exceed the suggested threshold of $c = 200$ and are clipped. Namely, the actual clipping error cannot be ignored, creating a gap between theory and practice.

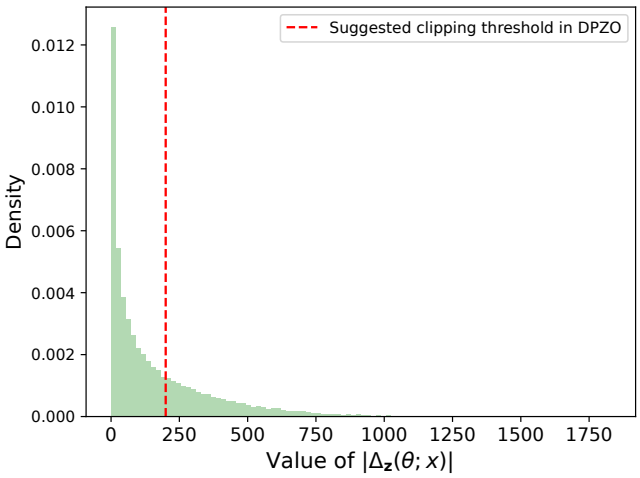


Figure 1: The empirical distribution (histogram) of $|\Delta_{\mathbf{z}}(\boldsymbol{\theta}; x)|$ computed for RoBERTa (355M) on the MNLI dataset. Sample values of $\Delta_{\mathbf{z}}(\boldsymbol{\theta}; x)$ are computed based on 512 perturbation vectors that are independently sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I}_d)$. Here, $\boldsymbol{\theta}$ refers to the model checkpoint taken midway through fine-tuning with *DPZero*. Similar patterns appear for other models (such as OPT), model checkpoints, and datasets.

Our analysis. To our knowledge, no follow-up study on differentially private zeroth-order fine-tuning (e.g. [63, 96]) has yet addressed this mismatch. Now that the clipping error cannot be ignored, we aim to quantify its impact on the convergence rate. Without loss of generality, we focus on one iteration t and fix the clipping threshold to c (with $c > 0$) and the noise multiplier to σ_m. Recall Section 2.3. Given a perturbation vector $\mathbf{z} \in \mathbb{R}^d$, *DPZero* first samples batch $\mathcal{B}_t$ of records, computes the scalar $\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)$ for each record x in $\mathcal{B}_t$, and then applies artificial clipping to obtain $\widehat{\Delta}_{\mathbf{z}}(\boldsymbol{\theta}_t; x)$. Next, $u \sim \mathcal{N}(0, \sigma_m^2 c^2)$ is injected to $\sum_{x \in \mathcal{B}_t} \widehat{\Delta}_{\mathbf{z}}(\boldsymbol{\theta}_t; x)$ and the model is updated as $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \frac{\left(\sum_{x \in \mathcal{B}_t} \widehat{\Delta}_{\mathbf{z}}(\boldsymbol{\theta}_t; x)\right) + u}{b} \cdot \mathbf{z}$.

We use $\mathcal{L}(\boldsymbol{\theta}_t)$ and $\nabla \mathcal{L}(\boldsymbol{\theta}_t)$ to denote the average loss and gradients computed on all records in input dataset $\mathcal{D}$. The convergence rate, i.e., expected loss decrease, is expressed as $\mathbb{E}[\mathcal{L}(\boldsymbol{\theta}_{t+1}) | \boldsymbol{\theta}_t] - \mathcal{L}(\boldsymbol{\theta}_t)$. We would want this difference to be negative when we update the model (i.e., the loss decreases).

We define the clipping error for the zeroth-order gradient

of record x, perturbation vector $\mathbf{z}$, and model $\boldsymbol{\theta}_t$ as

$$\text{err}(\boldsymbol{\theta}_t, x, \mathbf{z}) = \frac{\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x) - \widehat{\Delta}_{\mathbf{z}}(\boldsymbol{\theta}_t; x)}{\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)}. \tag{10}$$

Accordingly, the clipping error incurred in batch $\mathcal{B}_t$ is

$$\text{err}(\boldsymbol{\theta}_t, \mathcal{B}_t, \mathbf{z}) = \frac{\sum_{x \in \mathcal{B}_t} \Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x) - \sum_{x \in \mathcal{B}_t} \widehat{\Delta}_{\mathbf{z}}(\boldsymbol{\theta}_t; x)}{\sum_{x \in \mathcal{B}_t} \Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)}. \tag{11}$$

We are ready to present the convergence rate for *DPZero*.

Lemma 2 (Convergence rate for DPZero)**.** *Assume that* $\nabla^2 \mathcal{L}(\boldsymbol{\theta}_t) \preceq \mathbf{H}$ *where* $\mathbf{H} \succeq l \cdot \mathbf{I}_d$ *and* $tr(\mathbf{H})/\|\mathbf{H}\|_{op} \leq r$*. If the model is updated as in equation 9, then we have*

$$\begin{aligned}
&\mathbb{E}[\mathcal{L}(\boldsymbol{\theta}_{t+1}) \,|\, \boldsymbol{\theta}_t] - \mathcal{L}(\boldsymbol{\theta}_t) \\
&\leq -\eta \cdot (1 - \mathbb{E}[err(\boldsymbol{\theta}_t, \mathcal{B}_t, \mathbf{z})]) \cdot \|\nabla \mathcal{L}(\boldsymbol{\theta}_t)\|^2 && (12) \\
&\quad + \sqrt{3}\eta \cdot \mathbb{E}[(err(\boldsymbol{\theta}_t, \mathcal{B}_t, \mathbf{z})^2] \cdot \mathbb{E}[\|\nabla \mathcal{L}(\boldsymbol{\theta}_t; x)\|^2] && (13) \\
&\quad + \frac{\eta^2 lr}{2} \cdot c^2 \cdot \left(1 + \sigma_m^2\right). && (14)
\end{aligned}$$

To better understand Lemma 2, we compare it with the convergence rate for non-DP zeroth-order fine-tuning. In particular, under the same assumptions as in Lemma 2, if the model is updated as $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \frac{\sum_{x \in \mathcal{B}_t} \Delta_{\mathbf{z}}(\boldsymbol{\theta}; x)}{|\mathcal{B}_t|} \cdot \mathbf{z}$, but without artificial clipping and additive DP noises, then the expected loss decrease can be bounded as

$$\begin{aligned}
\mathbb{E}[\mathcal{L}(\boldsymbol{\theta}_{t+1}) \,|\, \boldsymbol{\theta}_t] - \mathcal{L}(\boldsymbol{\theta}_t) &\leq -\eta \cdot \|\nabla \mathcal{L}(\boldsymbol{\theta}_t)\|^2 && (15) \\
&\quad + \frac{\eta^2 lr}{2} \cdot \|\nabla \mathcal{L}(\boldsymbol{\theta}_t)\|^2. && (16)
\end{aligned}$$

Differences due to DP. Artificial clipping and the additive DP noise cause the differences between the convergence rates of non-DP and DP zeroth-order fine-tuning. In particular, comparing equation 12 with equation 15, we see that the clipping error reduces the loss decrease of non-DP ZO by a factor of $1 - \mathbb{E}[err(\boldsymbol{\theta}_t, \mathcal{B}_t, \mathbf{z})]$ while introducing another positive term related to $\mathbb{E}[\|\nabla \mathcal{L}(\boldsymbol{\theta}_t; x)\|^2]$, slowing down the convergence. One benefit of artificial clipping is to prevent $\|\nabla \mathcal{L}(\boldsymbol{\theta}_t)\|^2$ from being larger than c^2 (comparing equation 14 with equation 16). This, however, comes with the price of clipping error, which might slow down the overall convergence, as we have mentioned above. In particular, consider the extreme case of setting $c = 0$. In this case, the term $\frac{\eta^2 lr}{2} \cdot c^2 \cdot (1 + \sigma_m^2)$ disappears while the model update is always $\mathbf{0}$, and we cannot expect the model to converge. Finally, $\sigma_m^2 c^2$ is the variance of the additive DP noise (see equation 14), which may slow down the model convergence.

Now we see a *quantitative relationship* between the clipping error and the convergence rate. In particular, decreasing the scale of the clipping error would in general increase the scale of negative term in equation 12 while decreasing the scale of the positive term in equation 13, improving the overall convergence rate.

Lemma 2 is the first in the literature that connects the clipping error to the convergence rate for *DPZero*. As we have mentioned earlier, prior analyses assume that clipping happens very rarely, limiting the real-world applicability. An analysis that is similar to ours in the spirit was derived in a different context - training deep neural networks from scratch using first-order SGD with differential privacy [106] - which does not apply to our problem. We defer the detailed proof of Lemma 2 to the full technical report [12], where we have adopted the low-effective rank assumption, a standard practice for analyzing (large) language models [2, 44, 78, 108].

4.2 Motivation: Clipping Error Reduction

Before introducing the idea to improve the privacy-utility trade-off of *DPZero*, we first highlight the technical challenge.

Challenge. As we can see from Lemma 2, to improve the convergence rate of *DPZero*, the first thing that comes is to reduce the clipping error. However, this can only be achieved (so far) through setting a larger clipping threshold, i.e., a less restrictive clipping strategy. In particular, if we set $c = \infty$, then the clipping error becomes 0. However, as the additive DP noise is proportional to the clipping threshold c, if we set c to a larger value, the term $\frac{\eta^2 lr}{2} \cdot (1 + \sigma_m^2 c^2)$ also becomes larger, meaning that the error due to additive DP noises increases, potentially slowing down the convergence.

The above discussion highlights the tension between the clipping error and the additive noise: reducing one necessarily increases the other. In the literature, this phenomenon is commonly referred to as the *bias-variance trade-off*, which has been studied in various domains (e.g., see [18, 106] for deep learning applications, [3, 10, 35, 36] for database and statistical queries, and [33, 54] for theory). Nonetheless, it was not addressed in the context of fine-tuning LLMs with DP. Here, our goal is to reduce the clipping error (i.e., bias) without compromising for larger additive DP noises (i.e., variance). In what follows, we present our idea to solving this problem.

Bounding err$(\boldsymbol{\theta}_t, x, \mathbf{z})$ from above. Our goal is to reduce the clipping error without directly adjusting the clipping threshold to larger values. It is difficult to directly work on the clipping error with respect to the batch, i.e., err$(\boldsymbol{\theta}_t, \mathcal{B}_t, \mathbf{z})$, since we do not know the distribution of records in the dataset. Instead, we take a heuristic approach and try to reduce the clipping error with respect to a record x, written as err$(\boldsymbol{\theta}_t, x, \mathbf{z})$ (recall its definition from equation 10). The intuition is as follows. If we can reduce err$(\boldsymbol{\theta}_t, x, \mathbf{z})$ to some small value, that means the values of $\widehat{\Delta}_{\mathbf{z}}(\boldsymbol{\theta}_t; x)$ and $\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)$ are close. As a consequence, by taking the sum over all records from batch $\mathcal{B}_t$, err$(\boldsymbol{\theta}_t, \mathcal{B}_t, \mathbf{z})$ is also likely to be small. Still, err$(\boldsymbol{\theta}_t, x, \mathbf{z})$ depends on $\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)$, for which we also have very little knowledge. Instead, we try

to bound this value from above and then show how to reduce this upper bound.

For any clipping threshold $c > 0$, the clipped result $\widehat{\Delta}_{\mathbf{z}}(\boldsymbol{\theta}_t; x)$ is always the original $\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)$ times some scaling factor in $(0, 1]$. Hence, we have $1 - \frac{|\widehat{\Delta}_{\mathbf{z}}(\boldsymbol{\theta}_t;x)|}{|\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t;x)|} \leq \frac{|\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t;x)|}{|\widehat{\Delta}_{\mathbf{z}}(\boldsymbol{\theta}_t;x)|} - 1$, and we can show that the clipping error $\mathrm{err}(\boldsymbol{\theta}_t, x, \mathbf{z})$ satisfies

$$\mathrm{err}(\boldsymbol{\theta}_t, x, \mathbf{z}) \leq \frac{1}{c} \cdot \mathbb{E}[|\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)| \cdot \mathbf{1}\{|\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)| > c\}]. \quad (17)$$

Applying the Cauchy-Schwarz inequality and Chebyshev's inequality for $c^2 > \mathbb{E}\left[|\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)|^2\right]$, we can further bound

$$\mathrm{err}(\boldsymbol{\theta}_t, x, \mathbf{z}) \leq \frac{\mathbb{E}\left[|\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)|^2\right]}{c^2} = \frac{\mathrm{cov}\left[\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)\right]}{c^2}, \quad (18)$$

where we have used the law of total variance and the fact that $\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)$ is of mean 0 on $\mathbf{z}$ randomly sampled from the Gaussian distribution.

4.3 Idea: Aggregating ZO Estimates

Now we are ready to introduce our idea for improving *DPZero*. The key is to reduce $\mathrm{cov}\left[\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)\right]$, the deviation of $\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)$ each x.

To illustrate the idea, we focus on one specific iteration t without loss of generality. For each record x, instead of computing a single estimate for the gradient $\nabla \mathcal{L}(\boldsymbol{\theta}_t; x)$ using $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$, we query the model's losses on multiple $\mathbf{z}$'s and use the aggregate of the zeroth-order estimates as the approximation for the exact gradient. In particular, we first independently sample a set of K (with $K > 1$) random vectors from the distribution $\mathcal{N}(\mathbf{0}, \mathbf{I}_d)$. We denote the vectors as $\{\mathbf{z}_k\}_{k=1}^K$. Next, for each sampled record, we compute

$$\Delta_{\mathbf{z}_k}(\boldsymbol{\theta}_t; x) := \frac{\mathcal{L}(\boldsymbol{\theta}_t + \phi \mathbf{z}_k; x) - \mathcal{L}(\boldsymbol{\theta}_t - \phi \mathbf{z}_k; x)}{2\phi}, \quad (19)$$

for all $k = 1, \ldots, K$. Based on Proposition 1, each $\Delta_{\mathbf{z}_k}(\boldsymbol{\theta}_t; x) \cdot \mathbf{z}_k$ is an unbiased estimate for the true gradient. Hence, their aggregate average, expressed as $\frac{1}{K}\sum_{k=1}^K \Delta_{\mathbf{z}_k}(\boldsymbol{\theta}_t; x) \cdot \mathbf{z}_k$, is an unbiased estimate for the true gradient $\nabla \mathcal{L}(\boldsymbol{\theta}_t; x)$. We enforce differential privacy on this statistic.

Clipping the aggregate vector. Similar to *DPZero*, we see the random perturbation vectors $\{\mathbf{z}_k\}_{k=1}^K$ as public information. We focus on the K-dimensional aggregate vector

$$\boldsymbol{\Delta}_{\mathrm{agg}}(\boldsymbol{\theta}_t; x) = \frac{1}{K}(\Delta_{\mathbf{z}_1}(\boldsymbol{\theta}_t; x), \Delta_{\mathbf{z}_2}(\boldsymbol{\theta}_t; x), \ldots, \Delta_{\mathbf{z}_K}(\boldsymbol{\theta}_t; x)). \quad (20)$$

Each dimension independently follows the same distribution. The standard deviation of the norm of this K-dimensional aggregate vector is $\sqrt{K}$ times smaller than the standard deviation of original $\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)$ computed on a single perturbation vector. As a result, it now becomes easier to clip the individual aggregate vectors (compared with the original one-dimensional scalars) as large $\Delta_{\mathbf{z}_{k_1}}(\boldsymbol{\theta}_t; x)$ and small $\Delta_{\mathbf{z}_{k_2}}(\boldsymbol{\theta}_t; x)$ naturally offset each other to make the vector's norm more stable. Namely, it is less likely to observe an individual that is either too large (incurring a large clipping error) or too small (wasting privacy budgets). Rather than setting the clipping threshold to a large value to encompass possible outliers, we can set it to some moderate values and the clipping error reduces to some constant as K increases, as we will formally show later.

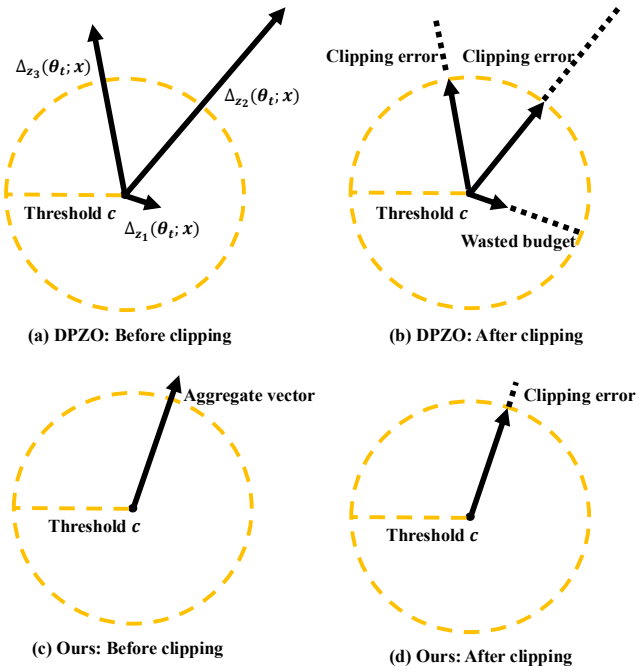


Figure 2: Illustration of our idea for clipping.

In Figure 2, we illustrate the advantage of our idea. Consider $K = 3$ with three perturbation vectors $\mathbf{z}_1$, $\mathbf{z}_2$, and $\mathbf{z}_3$, we first compute $\Delta_{\mathbf{z}_1}(\boldsymbol{\theta}_t; x)$, $\Delta_{\mathbf{z}_2}(\boldsymbol{\theta}_t; x)$, and $\Delta_{\mathbf{z}_3}(\boldsymbol{\theta}_t; x)$, as shown in the top-left corner. For illustration purposes, each one of them is pointing towards a different direction. Using the original *DPZero*, we might observe **i):** large clipping errors if the loss difference is much larger than the clipping threshold c, or **ii):** wasted privacy budgets if the loss difference is much smaller than c. To avoid these, however, is not easy. As we have mentioned, simply increasing the threshold c would introduce excessive DP noises (i.e., wasting privacy budget) whereas decreasing c would make the clipping errors even larger. Using our idea, the norm of the aggregate vector converges to some value (as we will formally show later), making it easier to achieve small clipping error without introducing excessive DP noises.

Comparison with [63]. Recent work [63] also considers using multiple independent zeroth-order estimates obtained at different sampled perturbation vectors. However, their solution does not resolve the issue of clipping error. In [63], for each $\mathbf{z}_k$, they apply artificial clipping to $\Delta_{\mathbf{z}_k}(\boldsymbol{\theta}_t; x)$ inde-

pendently. Clearly, the clipping error is not mitigated in their approach: as shown in our Figure 2, for each perturbation vector $\mathbf{z}_k$, the observed $\Delta_{\mathbf{z}_k}(\boldsymbol{\theta}_t;x)$ may be divergent, leading to a large clipping error or wasted privacy budget.

5 Our Solution: DP-AggZO

5.1 Algorithm

The complete procedure of our solution, called *DP-AggZO*, is outlined in Algorithm 1. As we have mentioned in Section 4.3, the main difference from *DPZero* is that we first sample a set of K (with $K > 1$) perturbation vectors from $\mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ independently, and then query the model to compute $\Delta_{\mathbf{z}_k}$ for all $\mathbf{z}_k, k = 1, \ldots, K$ (Lines 5-10 in Algorithm 1). Note that the computation of $\Delta_{\mathbf{z}_k}(\boldsymbol{\theta}_t;x)$'s are run in sequential orders. Hence, the memory consumption remains the same as *DPZero*.

To obtain a differentially private estimate for the sum of the (approximated) gradients over a batch, we first perform clipping for each individual record x's K-dimensional vector (Lines 11-13 in Algorithm 1)

$$\boldsymbol{\Delta}_{\text{agg}}(\boldsymbol{\theta}_t;x) = \frac{1}{K}(\Delta_{\mathbf{z}_1}(\boldsymbol{\theta}_t;x), \Delta_{\mathbf{z}_2}(\boldsymbol{\theta}_t;x), \ldots, \Delta_{\mathbf{z}_K}(\boldsymbol{\theta}_t;x)).$$

We clip the $\mathcal{L}_2$ norm of this vector to some pre-determined threshold c. The clipped vector is expressed as

$$\widehat{\boldsymbol{\Delta}}_{\text{agg}}(\boldsymbol{\theta}_t;x) = \frac{c}{\max(c, \|\boldsymbol{\Delta}_{\text{agg}}(\boldsymbol{\theta}_t;x)\|)} \cdot \boldsymbol{\Delta}_{\text{agg}}(\boldsymbol{\theta}_t;x). \quad (21)$$

We denote the k-th dimension of the clipped vector as $\widehat{\Delta}_{\text{agg},k}(\boldsymbol{\theta}_t;x)$. After clipping is done, for each dimension k, we inject Gaussian noise $v_k \sim \mathcal{N}(0, \sigma_m^2 c^2)$ (here σ_m is the noise multiplier) to the sum of the clipped vectors over the k-th dimension, i.e., $\sum_{x\in\mathcal{B}} \widehat{\Delta}_{\text{agg},k}(\boldsymbol{\theta}_t;x)$, obtaining

$$\widetilde{G} := \sum_{k=1}^{K} \Big(\sum_{x\in\mathcal{B}} \widehat{\Delta}_{\text{agg},k}(\boldsymbol{\theta}_t;x) + v_k \Big) \cdot \mathbf{z}_k \quad (22)$$

as the DP zeroth-order estimate for the gradient sum for batch $\mathcal{B}$ (Lines 14-15 in Algorithm 1). With learning rate set to η, the model is updated as

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \frac{\eta}{b} \cdot \widetilde{G}, \quad (23)$$

where b is some pre-determined targeted batch size (Line 16 in Algorithm 1).

5.2 Illustration of Reduced Clipping Error

We demonstrate how our solution is able to reduce the clipping error. The detailed proofs are deffered to the full version. Similar to *DPZero*, we first define the clipping error for our

Algorithm 1: DP-AggZO

Input: private dataset D of size n; total number of training steps T; subsampling rate q; learning rate η; initial (pre-trained) model parameters $\boldsymbol{\theta}_0$; perturbation scale ϕ; Gaussian noise multiplier σ_m; clipping threshold c; Laplace noise parameter ψ for perturbing n; number of random directions K.

1 Perturb the size of $\mathcal{D}$: $\tilde{n} = n + \text{Lap}(0, \psi)$.
2 Compute targeted batch size: $b = q\tilde{n}$.
3 **for** $t = 1 \ldots T$ **do**
4 Obtain batch $\mathcal{B}_t$ using Poisson sampling, where each record in $\mathcal{D}$ is sampled with probability q .
5 **for** $k = 1 \ldots K$ **do**
6 Sample $\mathbf{z}_k$ from the standard Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{I}_d)$. // d is the dimension of model parameters
7 Compute $\boldsymbol{\theta}_t^+ \leftarrow \boldsymbol{\theta}_t + \phi \cdot \mathbf{z}_k$.
8 Compute $\boldsymbol{\theta}_t^- \leftarrow \boldsymbol{\theta}_t - \phi \cdot \mathbf{z}_k$.
9 **for** $x \in \mathcal{B}_t$ **do**
10 Query the model: $\Delta_{\mathbf{z}_k}(\boldsymbol{\theta}_t;x) = \frac{\mathcal{L}(\boldsymbol{\theta}_t^+;x) - \mathcal{L}(\boldsymbol{\theta}_t^-;x)}{2\phi}$
11 **for** x *in batch* $\mathcal{B}_t$ **do**
12 Obtain K-dimensional vector $\boldsymbol{\Delta}_{\text{agg}}(\boldsymbol{\theta}_t;x) = \frac{1}{K}(\Delta_{\mathbf{z}_1}(\boldsymbol{\theta}_t;x), \ldots, \Delta_{\mathbf{z}_K}(\boldsymbol{\theta}_t;x))$.
13 Artificial clipping: $\widehat{\boldsymbol{\Delta}}_{\text{agg}}(\boldsymbol{\theta}_t;x) = \frac{\min(\|\boldsymbol{\Delta}_{\text{agg}}(\boldsymbol{\theta}_t;x)\|, c)}{c} \cdot \boldsymbol{\Delta}_{\text{agg}}(\boldsymbol{\theta}_t;x)$.
14 **for** *each dimension* k *in vector* $\widehat{\boldsymbol{\Delta}}_{agg}(\boldsymbol{\theta}_t;x))$ **do**
15 Noise injection: $\widetilde{\Delta}_{\text{agg},k}(\boldsymbol{\theta}_t;\mathcal{B}) = \big(\sum_{x\in B} \widehat{\Delta}_{\text{agg},k}(\boldsymbol{\theta}_t;x)\big) + \mathcal{N}(0, \sigma_m^2 c^2)$.
16 Update model parameters $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \frac{\eta}{b} \cdot \sum_{k=1}^{K} \widetilde{\Delta}_{\text{agg},k}(\boldsymbol{\theta}_t;\mathcal{B}) \cdot \mathbf{z}_k$.

Output: Model parameters θ.

DP-AggZO. For any record x, model parameter $\boldsymbol{\theta}_t$, and perturbation vectors $\{\mathbf{z}_k\}_{k=1}^K$, we define the clipping error as

$$err_{\text{agg}}(\boldsymbol{\theta}_t, x, \{\mathbf{z}_k\}_{k=1}^K) = \frac{\|\boldsymbol{\Delta}_{\text{agg}}(\boldsymbol{\theta}_t;x) - \widehat{\boldsymbol{\Delta}}_{\text{agg}}(\boldsymbol{\theta}_t;x)\|}{\|\boldsymbol{\Delta}_{\text{agg}}(\boldsymbol{\theta}_t;x)\|}. \quad (24)$$

With $\boldsymbol{\theta}_t$ and x fixed, this error depends on $\{\mathbf{z}_k\}_{k=1}^K$. In what follows, we show that this clipping error converges to some constant as we increase K. We use σ_x^2 to denote the variance of $\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t;x)$ on a randomly sampled $\mathbf{z}$. The core argument is that the norm of the K-dimensional vector $\boldsymbol{\Delta}_{\text{agg}}(\boldsymbol{\theta}_t;x)$ converges to $\frac{\sigma_x}{\sqrt{K}}$, formalized as follows.

Lemma 3. *[Bound on $\|\boldsymbol{\Delta}_{agg}(\boldsymbol{\theta}_t;x)\|$] Let $U > 1$ be some positive constant such that $|(\Delta_{\mathbf{z}_j}(\boldsymbol{\theta}_t;x))^2 - \mathbb{E}[(\Delta_{\mathbf{z}_j}(\boldsymbol{\theta}_t;x))^2]| < U$ almost surely for every record x, then for every record x,*

$$\Pr\Big[\|\boldsymbol{\Delta}_{agg}(\boldsymbol{\theta}_t;x)\| > \frac{\sigma_x}{\sqrt{K}} + \sqrt{\frac{3\sqrt{\log K}}{K\sqrt{K}}}\sqrt{U}\Big] \le \frac{1}{K^2}.$$

Now that the norm is bounded, we can show that given a clipping threshold $c = \beta_{\text{clip}} \cdot \frac{1}{\sqrt{K}}$ for some β_{clip}, with probability at least $1 - \frac{1}{K^2}$, the clipping error is also bounded:

$$err_{\text{agg}}(\boldsymbol{\theta}_t, x, \{\mathbf{z}_k\}_{k=1}^K) \leq \frac{\sigma_x}{\beta_{\text{clip}}} + \Theta\left(\left(\frac{\log K}{K}\right)^{\frac{1}{4}}\right). \quad (25)$$

When K is large, this upper bound converges to $\frac{\sigma_x}{\beta_{\text{clip}}}$. We note that in the above derivation, we did not utilize the Chebyshev's inequality or Markov's inequality which requires that β_{clip} to be larger than σ_x. Instead, we rely on the concentration bound for Bernstein's inequality, which basically states that the tail of $\|\boldsymbol{\Delta}_{\text{agg}}(\boldsymbol{\theta}_t; x)\|$ becomes lighter as K increases.

Comparison with *DPZero*. With $K = 1$ (i.e., using the original *DPZero*), it is difficult to bound the clipping error to a small value. In particular, if we assume that $\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)$ follows a zero-mean Gaussian distribution, then a sampled value has at least around 13% probability to be larger than $1.5\sigma_x$, giving a clipping error of $\frac{1.5\sigma_x}{\beta_{\text{clip}}}$. Alternatively, the distribution of $\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)$ may not be as concentrated as a Gaussian: if it follows a zero-mean Laplace distribution (which has a heavier tail), then a sampled value has at least around 30% probability to be larger than $2\sigma_x$, giving a clipping error of $\frac{2\sigma_x}{\beta_{\text{clip}}}$. Using our method, the clipping error is always upper bounded by $\frac{\sigma_x}{\beta_{\text{clip}}}$, for any record x as long as K is large enough.

Simulation results. We provide simulation results to back up our analysis. The setup is as follows. Given model parameters $\boldsymbol{\theta}$, we first sample a batch of records (denoted as $\mathcal{B}_t$) and K perturbation vectors $\{\mathbf{z}_k\}_{k=1}^K$ independently. Next, we compute $\Delta_{\mathbf{z}_k}(\boldsymbol{\theta}; x)$ for each record $x \in \mathcal{B}_t$. After this, we clip the aggregate K-dimensional vector $\boldsymbol{\Delta}_{\text{agg}}(\boldsymbol{\theta}_t; x)$ using threshold $c = \frac{\beta_{\text{clip}}}{\sqrt{K}}$ for each record x, with β_{clip} set to 200 (the suggested value in [113]). We measure the maximum clipping error of any record $x \in \mathcal{B}_t$, written as

$$err_{\text{max individual}} := \max_{x \in \mathcal{B}_t} \frac{\|\boldsymbol{\Delta}_{\text{agg}}(\boldsymbol{\theta}_t; x) - \widehat{\boldsymbol{\Delta}}_{\text{agg}}(\boldsymbol{\theta}_t; x)\|}{\|\boldsymbol{\Delta}_{\text{agg}}(\boldsymbol{\theta}_t; x)\|}, \quad (26)$$

and the overall clipping error of batch $\mathcal{B}_t$, written as

$$err_{\text{batch } \mathcal{B}_t} := \frac{\|\sum_{x \in \mathcal{B}_t} \boldsymbol{\Delta}_{\text{agg}}(\boldsymbol{\theta}_t; x) - \sum_{x \in \mathcal{B}_t} \widehat{\boldsymbol{\Delta}}_{\text{agg}}(\boldsymbol{\theta}_t; x)\|}{\|\sum_{x \in \mathcal{B}_t} \boldsymbol{\Delta}_{\text{agg}}(\boldsymbol{\theta}_t; x)\|}. \quad (27)$$

We repeat the above process with K ranging from 1 to 256. When $K = 1$, we are simulating the original *DPZero*. The results are shown in Figure 3. As K increases, both the maximum clipping error of individuals and the clipping error converge to smaller values. With $K = 1$, the batch clipping error could exceed 1 (i.e., the model could be updated opposite of the original intended direction). On the other hand, as K increases, this error steadily decreases to some value below 0.5. In addition to that, we also note the correlation between the maximum individual clipping error and the batch clipping error, which backs up our heuristic mentioned in Section 4.2; that is, to reduce the batch clipping error, we target at reducing the maximum possible individual clipping error. Similar patterns appear for other models, datasets, and clipping thresholds. We will see how the clipping error fluctuates during the fine-tuning process in Section 6.

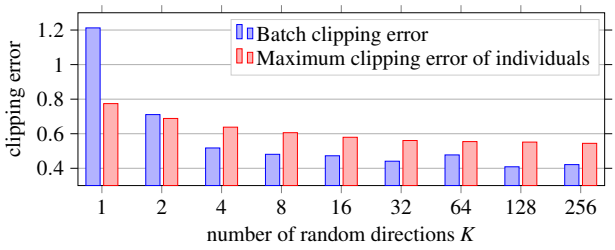


Figure 3: Illustration of the clipping errors with respect to individuals and a batch under different K's, computed on a model checkpoint during the fine-tuning process of RoBERTa (355M) on MNLI dataset.

5.3 Analysis

In this subsection, we present the convergence and privacy guarantees for our DP-AggZO.

Lemma 4 (Convergence rate for DP-AggZO). *Assume that for all $\boldsymbol{\theta}$, we have $\nabla^2 \mathcal{L}(\boldsymbol{\theta}) \preceq \mathbf{H}$ where $\mathbf{H} \preceq l \cdot \mathbf{I}_d$ and $tr(\mathbf{H})/\|\mathbf{H}\|_{op} \leq r$. Then, for clipping threshold $c = \beta_{clip} \cdot \frac{1}{\sqrt{K}}$ such that the model is updated as in equation 23, we have*

$$\mathbb{E}[\mathcal{L}(\boldsymbol{\theta}_{t+1})|\boldsymbol{\theta}_t] \leq -\eta\left(1 - \frac{\mathbb{E}_x[\sigma_x]}{\beta_{clip}} - o(1)\right) \cdot \|\nabla \mathcal{L}(\boldsymbol{\theta}_t)\|^2 \quad (28)$$

$$+\eta \cdot \sqrt{3}\left(1 - \frac{\mathbb{E}_x[\sigma_x]}{\beta_{clip}} - o(1)\right) \cdot \frac{\mathbb{E}[\|\nabla \mathcal{L}(\boldsymbol{\theta}_t; x)\|^2]}{\sqrt{K}} \quad (29)$$

$$+\frac{\eta^2 \cdot lr}{2} \cdot \frac{\beta_{clip}^2}{K} + \frac{\eta^2 \cdot lr}{2} \sigma_m^2 \cdot \beta_{clip}^2. \quad (30)$$

The proof of Lemma 4 is similar to Lemma 2. The main difference is that we replace the zeroth-order estimate computed on a single random direction specified by $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ with a set of zeroth-order estimates computed on $\{\mathbf{z}_k\}_{k=1}^K$ where each $\mathbf{z}_k$ is independently sampled from the same distribution. We note that the $o(1)$ error comes from the error that $\|\boldsymbol{\Delta}_{\text{agg}}(\boldsymbol{\theta}_t; x)\|$ deviates from its high-probability bound (see Lemma 3), which converges to 0 as K increases.

The privacy cost of *DP-AggZO* comes from two parts. The first part is due to a zero-mean Laplace noise with scale parameter ψ for perturbing the input size $|\mathcal{D}|$, from which we then obtain the targeted batch size. The second part is due to the additive Gaussian noises applied to the ZO estimates computed on a random subset of the input obtained with Poisson sampling. The overall RDP guarantee is then entailed from

the composition and subsampling lemmas (see, e.g., [1, 71] for more details). In our experiments, the privacy cost for releasing a DP estimate for $|\mathcal{D}|$ consumes 5% of the overall privacy cost, and the rest of the privacy cost is used for releasing the ZO estimates for T model updates.

Lemma 5 (RDP guarantee for DP-AggZO)**.** *Running DP-AggZO for T iterations with a Poisson sampling rate of q, Gaussian noise multiplier σ_m for perturbing the ZO estimates, and a zero-mean Laplace noise with scale parameter ψ for perturbing $|\mathcal{D}|$, satisfies (α, ε)-RDP for $\alpha \in \mathbb{Z}, \alpha \geq 2$ with*

$$\varepsilon = \frac{1}{\alpha-1} \cdot \log\left(\frac{\alpha}{2\alpha-1}\exp\left(\frac{\alpha-1}{\psi}\right) + \frac{\alpha-1}{2\alpha-1}\exp\left(-\frac{\alpha}{\psi}\right)\right) + \frac{T}{\alpha-1} \cdot \log\Big((1-q)^{\alpha-1}(\alpha q - q + 1) + \sum_{h=2}^{\alpha}\binom{\alpha}{h}(1-q)^{\alpha-h}q^h e^{(h-1)\varepsilon(h)}\Big),$$

where $\varepsilon(h) = \frac{h}{2\sigma_m^2}$ for $h \in \mathbb{Z}, h \geq 2$.

The corresponding (ε, δ)-DP can be obtained accordingly by conversion rules (see [19]).

6 Experiments

In this section, we evaluate the performance of our *DP-AggZO* for fine-tuning pre-trained language models.

6.1 Setup

Models. Following prior work [63, 113], we conduct experiments on open-sourced pre-trained models, including the masked language model RoBERTA-Large (355M) [61] and the auto-regressive models OPT-1.3B and OPT-6.7B [115].

Datasets. We fine-tune on a wide range of datasets and tasks. For RoBERTa-Large, we fine-tune the model on six datasets for different classification tasks, including SST-2 and SST-5 [89] for sentiment analysis (i.e., determine if the given text is positive/negative), SNLI [14], MNLI [104], and RTE [31] for natural language inference (i.e., determine if the given premise and hypothesis are in the relationship of entailment/neutral/contradiction), and TREC [101] for topic assignment (i.e., determine which topic the given question falls under).

For larger models OPT-1.3B and OPT-6.7B that require more resources to fine-tune, we focus on one classification task SST-2 and one generation task SQuAD [82] (the fine-tuned model answers to a given question containing relevant contexts). We follow the data split of the previous work [116] and compare *DP-AggZO* against the following baselines.

DPAdamW. *DP-AdamW* [59], is based on the classic first-order method *DPSGD* that perturbs the sum of (clipped) gradients computed over a batch of records with additive multi-dimensional Gaussian noises [1]. The difference is that *DP-AdamW* adopts the optimizer AdamW [64] when performing the model updates. *DP-AdamW* is regarded as the state-of-the-art method for DP fine-tuning, as it achieves the best privacy-utility trade-off.

DPZero. *DPZero* [96, 113] replaces the first-order gradient with a zeroth-order approximation. This approximation is obtained as the result of a random vector sampled from the model's parameter space times a scalar that is an approximation to the model's directional derivative along the sampled vector. As reviewed in Section 2.3, additive DP noises are injected to the sum of the (clipped) scalars computed over a batch of records.

Non-DP baselines. We also include non-DP baselines for reference, where the models are fine-tuned to convergence (with no privacy budget limitation). The non-DP version of *DP-AdamW* is referred to as *AdamW*, which adopts the AdamW optimizer to the standard mini-batch SGD. *AdamW* is also seen as the upper bound for the model performance with DP. The non-DP version of *DPZero* is referred to as *MeZO* [68].

Zeroshot. *Zeroshot* directly tests the pre-trained model on the test dataset without fine-tuning, providing the strongest privacy protection for the fine-tuning data. *Zeroshot* serves as the lower bound for DP fine-tuning.

Other first-order DP fine-tuning methods, such as DP-LoRA and DP-Prefix Tuning [59, 110], are not included, as they often yield similar or inferior utility compared to *DP-AdamW* under the same privacy constraints. Overall, *DP-AdamW* is the main target that we aim to outperform.

Hyperparameters. For our method *DP-AggZO*, we consider two choices of K for each task. On the RoBERTa (355M) model, we set K to 64 and 256. On the larger models OPT-1.3B and OPT-6.7B, we set K to 16 and 64 to save the computation time. We test all baselines and our method under different hyperparameters (varying the subsampling rate, learning rate, and the clipping threshold). We report the best performance for each method on each task, following the convention in the literature. For detailed hyperparameters and the search grid, please refer to the technical report [12].

Evaluation metric. The performance is measured as the accuracy/F1 score (shown in %) on the test dataset for each task. Regarding DP, we focus on the (ε, δ)-DP framework with $\varepsilon = 2$ and 6 with $\delta = 10^{-5}$, as is the standard for evaluating DP algorithms (e.g., see [1, 32] and prior work [113]). Due to the scale of the experiments, we have mostly tested the performance on one random seed. For RoBERTa (355M), the random seed is fixed to 42; for OPT models (1.3B and 6.7B), the random seed is fixed to 0 (both are the default setups in the implementations of prior work [68, 116]). Cursory experiments on other seeds show that the standard deviations are around 1%. Hence, we have omitted those results. Our experiments are run on the H20 GPU with 96 GB of memory.

6.2 Main Results

We list the performance of different methods in Tables 1 and 2, including those without any DP constraint for reference. As a sanity check, the *Zero-shot* baseline is consistently the worst under all setups, motivating fine-tuning over sensitive data to improve the utility.

DP-AggZO outperforms DPZero. On all tasks and models, *DP-AggZO* significantly outperforms *DPZero*, setting a new benchmark for differentially private zeroth-order fine-tuning. Notably, the relative improvement increases as we increase K for *DP-AggZO*, which is aligned with our theoretical analysis. For the RoBERTa (355M) model, for example, *DP-AggZO* improves the performance of *DPZero* from 67.4% to 76.4% with $K = 64$, and further improves to 78.2% with $K = 256$, on the MNLI dataset under the DP constraint of $\varepsilon = 6, \delta = 10^{-5}$. Similarly, under the same DP constraint but for the OPT-6.7B model with the SQuAD dataset, *DP-AggZO* improves the performance of *DPZero* from 80.1% to 83.3% with $K = 16$, and further improves to 84.0% with $K = 64$.

DP-AggZO sometimes outperforms DP-AdamW. *DP-AggZO* also consistently outperforms the state-of-the-art *DP-AdamW* for RoBERTa (355M) model (Table 1). For example, on TREC dataset, when the DP constraint is $\varepsilon = 6, \delta = 10^{-5}$, *DP-AggZO* outperforms *DP-AdamW* by around 2%. On OPT models (Table 2), the advantage of memory savings for zeroth-order methods is more notable. In particular, first-order methods *AdamW* and *DP-AdamW* fail on OPT-6.7B, due to its large consumption of memory, whereas our *DP-AggZO* performs the best among the zeroth-order methods. For the OPT-1.3B model, *DP-AdamW* and *DP-AggZO* perform pretty close under the same DP constraints.

Non-DP baselines. Without DP, AdamW (a variant of SGD) achieves the best overall performance. We attribute this to the accurate gradient computation of the first-order method (recall that all zeroth-order methods computes approximations of the exact gradient). There are no notable differences between *MeZO* and the non-DP versions of our *DP-AggZO* as there is no privacy constraint - the zeroth-order gradients are not clipped in the first place and the models are trained to convergence (this is also aligned with the observations from [68]). The gaps between *MeZO* and its DP version *DPZero* are significantly larger than those between the non-DP and DP versions of our solution. Take the RoBERTa model on TREC dataset as an example (Table 1), the gap due to DP for *DPZero* is around 7% (when $\varepsilon = 6, \delta = 10^{-5}$) and 11% (when $\varepsilon = 2, \delta = 10^{-5}$). On the contrary, for *DP-AggZO* with $K = 64$, the gap due to DP is only 0.8% (when $\varepsilon = 6, \delta = 10^{-5}$) and 3.2% (when $\varepsilon = 2, \delta = 10^{-5}$). This improvement is due to the reduction of clipping error in *DP-AggZO*. Next, we provide possible explanations for the observed results.

Visualizing the clipping error. In Figure 4, we plot the relative clipping error of the sampled batch in each iteration for *DP-AdamW* and our *DP-AggZO* with $\varepsilon = 2$ and $\delta = 10^{-5}$. For *DP-AdamW*, we test clipping threshold of $50, 100$, and 200. For *DP-AggZO*, we set $K = 16$ and 256. We also include the setting with $K = 1$, which corresponds to the *DPZero* baseline. With different K's, the clipping threshold for *DP-AggZO* is set to $\frac{\beta_{\text{clip}}}{\sqrt{K}}$ (so that the additive DP noise remains a constant), with $\beta_{\text{clip}} = 100$ or 200. That is, for $K = 16$, we consider clipping threshold $c = 25$ and 50; for $K = 256$, we consider clipping threshold of $c = 6.25$ and $c = 12.5$. We show the clipping errors that occurred during the first 200 iterations out of $T = 1000$ for readability. Similar patterns appear for the rest of the fine-tuning process, other models, datasets, and privacy constraints.

Comparison with *DPZero*. First, *DP-AggZO* outperforms *DPZero* due to the reduced clipping error. According to our analysis in Section 5, for *DPZero*, it is difficult to control the scale of the zeroth-order gradient that we need to apply artificial clipping, leading to large and unstable clipping error. We back up this argument with empirical evidence. First and foremost, in Figure 4 (b), we observe that even under the suggested clipping thresholds of [113], the clipping error can be as large as 6, which means that the sum of the clipped zeroth-order gradients is in the opposite direction to the sum of the "unclipped" zeroth-order gradients in some iterations, contributing to the unsatisfactory privacy-utility trade-offs. In addition, as we increase the clipping threshold from 100 to 200, the clipping error reduces. However, as we have mentioned, this comes at the cost of additional DP noises due to the increased threshold. In *DP-AggZO*, with $K > 1$, we consider clipping threshold $c = \frac{\beta_{\text{clip}}}{\sqrt{K}}$. With β_{clip} fixed, as we increase K from 1 to 16 and then to 256, the clipping error reduces while the additive DP noise remains a constant (i.e., $\sigma_m^2 \beta_{\text{clip}}^2$), ultimately improving the convergence rate under same privacy constraints.

Comparison with DP-AdamW. *DP-AggZO* outperforms *DP-Adam*. The smaller clipping error of *DP-AggZO* contributes to this improvement, as we show in Figure 4 (e.g., comparing the clipping error of *DP-Adam* with $c = 200$ and that of *DP-AggZO* ($K = 256$) with $c = 200/\sqrt{K} = 12.5$). Again, similar to *DPZero*, directly increasing c for *DP-AdamW* does not solve the problem completely, as larger clipping thresholds in turn require larger DP noises injected, slowing down the model convergence - the positive term $\sigma_m^2 c^2 \cdot lr$ increases with c. We note that the gap between *DP-AdamW* and *DP-AggZO* is less notable on OPT-1.3B, compared with the smaller model RoBERTa (355M). The reason is that as the model's size becomes larger, its spectrum exhibits a more heavy-tailed distribution, which lead to smaller effective rank (i.e., smaller r) (e.g., see [69,93]). As a result, using larger clipping thresholds for *DP-AdamW* and introducing more DP noises to the model updates may not have a devastating impact on its performance (as the relative increase on $\sigma_m^2 c^2 \cdot lr$ is less significant compared to RoBERTa (355M)), reducing its performance gap

Table 1: Test performance on RoBERTa (355M). The best result is highlighted in bold.

Privacy constraint	**Algorithm**	—Sentiment—		—Natural Language Inference—			—Topic—
		SST-2	SST-5	SNLI	MNLI	RTE	TREC
$\varepsilon = 2, \delta = 10^{-5}$	DP-AdamW	91.7	47.5	74.6	73.5	72.8	91.6
	DPZero	91.8	47.1	73.6	62.7	70.4	83.8
	DP-AggZO ($K = 64$)	92.0	50.5	77.8	74.1	72.9	92.0
	DP-AggZO ($K = 256$)	**92.7**	**51.3**	**80.4**	**75.0**	**74.4**	**92.8**
$\varepsilon = 6, \delta = 10^{-5}$	DP-AdamW	92.4	49.0	81.5	76.3	77.3	92.8
	DPZero	92.2	49.3	77.8	67.4	71.9	87.6
	DP-AggZO ($K = 64$)	92.9	51.3	82.5	76.4	77.3	94.6
	DP-AggZO ($K = 256$)	**93.2**	**51.5**	**82.7**	**78.2**	**77.6**	**95.0**
Non-DP	AdamW	93.1	**56.6**	**86.4**	**81.4**	**83.6**	**95.9**
	MeZO	92.7	50.8	84.3	79.8	80.0	95.4
	DP-AggZO ($K = 64$)	93.4	51.6	84.9	79.9	80.5	94.8
	DP-AggZO ($K = 256$)	**94.0**	51.9	86.2	80.5	80.8	95.8
Perfect privacy	Zero-Shot	79.0	35.5	50.2	48.8	51.4	32.0

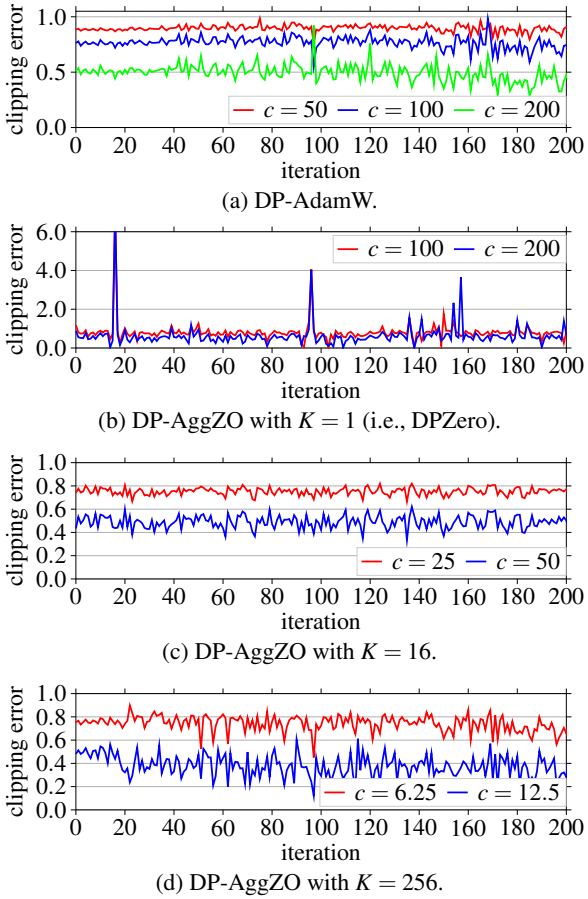
Table 2: Test performance on OPT-1.3B and OPT-6.7B. The best result is highlighted in bold. OOM stands for out of memory.

Privacy constraint	**Algorithm**	SST-2		SQuAD	
		OPT-1.3B	OPT-6.7B	OPT-1.3B	OPT-6.7B
$\varepsilon = 2, \delta = 10^{-5}$	DP-AdamW	91.0	OOM	78.0	OOM
	DPZero	86.6	92.7	72.3	78.5
	DP-AggZO ($K = 16$)	90.8	93.8	76.3	82.9
	DP-AggZO ($K = 64$)	**91.2**	**94.2**	**78.3**	**83.4**
$\varepsilon = 6, \delta = 10^{-5}$	DP-AdamW	91.3	OOM	79.2	OOM
	DPZero	88.2	92.9	74.2	80.1
	DP-AggZO ($K = 16$)	91.3	94.6	77.7	83.3
	DP-AggZO ($K = 64$)	**91.4**	**94.7**	**79.4**	**84.0**
Non-DP	AdamW	**93.6**	OOM	**81.5**	OOM
	MeZO	91.1	93.8	78.1	83.5
	DP-AggZO ($K = 16$)	92.6	94.6	78.9	84.2
	DP-AggZO ($K = 64$)	93.4	**95.5**	79.6	**85.4**
Perfect privacy	Zero-Shot	53.6	61.2	26.8	36.5

with *DP-AggZO*.

Comparison with the results in [59]. The performance of *DP-Adam* on MNLI and SST-2 reported here is different than [59]. The reason is that in our setting, for each dataset, we follow prior work [113] and sample 512 data points for each class from the original training dataset, resulting in 1024 and 1536 training data points in total for SST-2 and MNLI, respectively. This setup is different from and more difficult than [59], which uses the *full dataset* for training. Our setup better reflects real-world scenarios with limited data and is more challenging for enforcing DP. Hence, the performance of *DP-Adam* reported here is lower than [59]. Our reported numbers match those from [113].

Memory usage and computation. The memory usage of *DP-AggZO* on RoBERTa (355M) is provided in Table 3. For *DP-Adam*, to save the memory consumption, we process one data point at each time. Even under this scenario, we note that the memory cost of *DP-AdamW* is still much larger than *DPZero* (the former runs of out memory on OPT-6.7B model). This is attributed by two reasons. First, in zeroth-order optimization, we only need to cache the forward activations and the random perturbation vector during the forward pass through each layer. This cost can be as small as w, where w is the maximum layer width of the network. Our *DP-AggZO*, regardless of K (i.e., the number of random directions), always consumes the same memory as *DPZero*, since in our implementation, we query



(a) DP-AdamW.

(b) DP-AggZO with $K = 1$ (i.e., DPZero).

(c) DP-AggZO with $K = 16$.

(d) DP-AggZO with $K = 256$.

Figure 4: Clipping error comparison for different methods.

the zeroth-order loss difference on different random directions in a sequential manner (mentioned in Section 5). For first-order methods that involves backpropagation, all intermediate activation functions and gradients are cached. Combined with the gradient itself, gradient-based methods consume at least *three times more* memory than ZO methods. Besides, to run the AdamW optimizer, we also need to store the the first and second-order momentums and the adjusted gradients, leading to much larger overall memory usage [65].

Compared with *DPZero*, *DP-AggZO* requires more computation per update, since we query the model's losses over K random perturbation vectors for each record ($K > 1$). However, due to a faster convergence rate, *DP-AggZO* reaches a higher utility in fewer iterations. Overall, the additional computation cost of *DP-AggZO* is only 2 to 3 times more for $K = 64$ (*DP-AggZO* takes a few hundreds update to converge while *DPZero* takes thousands or even tens of thousands). Consider the much improved utility of *DP-AggZO* under the same privacy constraints, we believe this additional computation cost is reasonable to price to pay.

The small privacy budget regime. We also consider the DP constraint of $\varepsilon = 0.5$ and $\varepsilon = 1$ with $\delta = 10^{-5}$. On RoBERTa (355M), we consider one dataset for each task, including SST-2, MNLI, and TREC. The results are shown in Tabel 4. The improvement of our *DP-AggZO* over *DPZero* remains significant on MNLI and TREC and is less notable on SST-2. We suspect that the reason is that SST-2 is a binary classification task for sentiment analysis, which is easier than the other two tasks and less sensitive to clipping error.

Next, we provide some ablation studies on *DP-AggZO*.

Effect of K. In general, larger K leads to better performance for *DP-AggZO*. We demonstrate this phenomenon on the MNLI dataset using the RoBERTa (355M) model. We fix the subsampling rate to 0.0416, which leads to roughly 64 records in the sampled batch in each iteration. We fix $T = 1000$ iterations and vary K from $1, 4, 16, 64, 256, 512$, and 1024 (when $K = 1$, it is equivalent to the original *DPZero*). For each K, we test clipping thresholds from $\{0.5, 1, 2, 5, 25, 50, 100, 200\}$ and report the best accuracy. From Figure 5, it is clear that increasing K leads to improved performance; and the improvement becomes less notable when K is large enough (e.g.,

Table 3: Peak memory usage of DP-AdamW, DPZero, and DP-AggZO, tested on a H20 GPU. **The usage of DP-AggZO with different K's is identical.** See the full technical report version for OPT models.

Model	Dataset	Algorithm	Memory
RoBERTa (355M)	SST-2	DP-AdamW	**10.73** GiB
		DPZero	2.81 GiB
		DP-AggZO	2.81 GiB
RoBERTa (355M)	MNLI	DP-AdamW	**11.55** GiB
		DPZero	3.03 GiB
		DP-AggZO	3.03 GiB
RoBERTa (355M)	TREC	DP-AdamW	**11.41** GiB
		DPZero	2.67 GiB
		DP-AggZO	2.67 GiB

Table 4: Test performance on RoBERTa (355M) under the small privacy budget regimes, with $\varepsilon = 0.5$ and 1, and δ fixed to 10^{-5}. The best result is highlighted in bold.

Privacy	Algorithm	—Dataset—		
		SST-2	MNLI	TREC
$\varepsilon = 0.5$, $\delta = 10^{-5}$	DP-AdamW	90.6	62.0	77.6
	DPZero	90.3	57.1	73.8
	DP-AggZO ($K = 64$)	90.6	63.2	90.6
	DP-AggZO ($K = 256$)	**90.9**	**63.4**	**91.2**
$\varepsilon = 1$, $\delta = 10^{-5}$	DP-AdamW	91.6	68.0	85.0
	DPZero	91.2	60.5	82.6
	DP-AggZO ($K = 64$)	91.7	68.9	91.2
	DP-AggZO ($K = 256$)	**91.9**	**70.3**	**92.0**

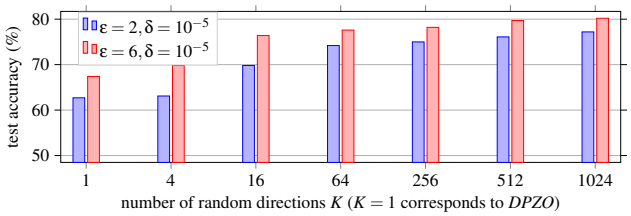


Figure 5: Performance of DP-AggZO on MNLI dataset using model RoBERTa (355M), with different K's.

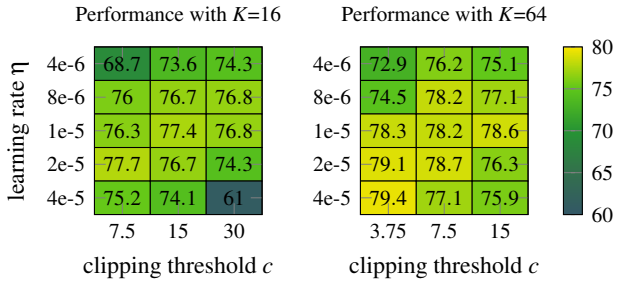


Figure 6: Performance of DP-AggZO ($K = 16, 64$) on SQuAD dataset using model OPT-1.3B, with varying learning rate and clipping threshold, and privacy fixed to $\varepsilon = 6, \delta = 10^{-5}$.

beyond 64), aligning with our theoretical analysis.

Effect of clipping threshold and learning rate. The clipping threshold c depends on K. In general, larger K needs a smaller clipping threshold since the expected $\mathcal{L}_2$ norm of the aggregated K-dimensional vector (recall equation 20) decreases as K increases. When the clipping threshold is small, we might in turn choose a larger learning rate η so that the model converges faster. This is because smaller clipping thresholds, along with larger K, lead to smaller variances in the model updates (without scaling with the learning rate). In this case, we should make the learning rate larger so that the expected loss decrease is more significant. This is confirmed by Figure 6. Comparing the performance of $K = 16$ with $K = 64$, the optimal clipping threshold in the search grid is smaller for the latter. In addition, when K is fixed, larger learning rates work better on smaller clipping thresholds, and vice versa. For example, when $K = 64$, the best performance is achieved at learning rate $\eta = 4e-5$ when the $c = 3.75$. On the other hand, when the clipping threshold $c = 15$, the best performance is achieved at $\eta = 1e-5$. Overall, if the primary focus is to achieve a better utility-privacy trade-off, then we recommend using large K (e.g., $K = 64$) associated with small clipping thresholds and large learning rates.

7 Related Work

Fine-tuning with DP. Fine-tuning large scale models using differential privacy in general adopts the same techniques for training DP models from scratch. We refer interested readers to [1, 18, 27, 42, 59, 77, 79, 97, 106, 110] for further read. As the sizes of models continually grow, some recent research studies on the computation and memory efficiency of DP methods, e.g., see [16, 17, 59, 110]. The potential privacy issue of fine-tuning over models pre-trained on *public data* is an open question [98], and is beyond the scope in this work.

DP zeroth-order methods. ZO for fine-tuning large models [68] was first adopted to the DP literature by [113] in their workshop version. Later on, [63, 96] follow up on this direction and propose different variants, demonstrating the potential and applicability of DPZO methods. The difference in our work is that we target at reducing the negative impact on the model's convergence due to artificial clipping for ZO. We accomplish this through algorithmic-level modifications and provide theoretical analysis to back up our design. Recent work [114] presents an algorithm with asymptotic improvements over existing DPZO methods, from a purely theoretical perspective, which assumes that the loss function is Lipschitz. As we have mentioned in Section 4.1, this assumption may not be valid in general, including LLM fine-tuning.

Without the constraint of differential privacy, using multiple independent zeroth-order estimates does not yield better utility under the same computation costs, as reported in [68]. We focus on improving the model's utility with a fixed differential privacy constraint. This is achieved by reducing the clipping error for releasing the aggregate of independent ZO estimates, at the cost of slightly more computation. As we have mentioned, directly incorporating multiple independent ZO estimates to *DPZO* [63] does not lead to significantly better utility under the same privacy constraints.

Clipping in DP. Clipping has been extensively studied in the DP literature [4, 52, 60, 83] and particularly for gradient clipping in the context of model training, ranging from traditional convolutional neural networks [1, 79], residual networks [32, 106], to transformers [59, 110]. Other than studying the bias-variance trade-off introduced by clipping (e.g., see [52, 106]), other related interesting problems include memory-efficient clipping for large model [16], optimizing computation for specific architectures [17], finding the optimal clipping threshold in deep learning/mean estimation [4, 5, 18, 32, 52, 116], and non-uniform clipping strategies [58, 105]. Finally, as we have mentioned in Section 4.2, existing clipping strategies for *DPSGD* do not directly apply to zeroth-order optimization, motivating this work.

8 Conclusion

In this work, we study the problem of fine-tuning LLMs under differential privacy. We propose *DP-AggZO*, a zeroth-order optimization algorithm with DP. *DP-AggZO* mitigates the error due to artificial clipping when enforcing DP over zeroth-order estimates for gradients. We provide rigorous analysis

and comprehensive experiments to validate its performance, which is significantly better than prior *DPZero* and even outperforms the state-of-the-art *DP-AdamW* in certain scenarios.

There is still room for improving DP fine-tuning, from the aspects of saving computation and memory cost while reducing the clipping error. A more efficient implementation of our method could also be a low-hanging fruit.

Acknowledgments

This work was supported by the Ministry of Education, Singapore, under Tier-2 Grant MOE-000761-01. We thank the anonymous shepherd and the reviewers for their constructive comments. We thank Hongyan Chang for coding advice, Yuan Qiu for discussions on zeroth-order optimization, and Yuexiang Xie for generously providing support on the computation clusters. We also thank Liang Zhang for discussions on the implementation details of DPZero.

9 Ethics Considerations

After carefully reviewing the ethics guidelines, we believe our research was conducted in an ethical manner. We follow the four principles in the Menlo report, "Beneficence", "Respect for Persons", "Justice", and "Respect for Law and Public Interest".

Beneficence. Our work aims to advance memory-efficient methods for differentially private (DP) fine-tuning of large language models (LLMs). Differential privacy is a crucial technique for safeguarding data privacy and mitigating the risks associated with LLMs. Existing DP fine-tuning methods either require substantial memory resources, forcing researchers and practitioners with limited hardware to rely on untrusted cloud providers, potentially exposing sensitive data, or lead to unsatisfactory model utility, forcing people to forfeit privacy protections. We propose a memory-efficient solution that achieve DP and high model utility at the same time, enabling DP fine-tuning on local devices with constrained computational resources and making privacy-preserving research and applications more accessible to a broader audience.

Respect for Persons. In our research, we adhere to this principle by ensuring that our experiments exclusively use publicly available datasets and open-sourced models. This approach eliminates the involvement of individuals whose autonomy or consent would otherwise need to be considered, and ensures that no personal data is exposed or at risk. Furthermore, our use of publicly available resources respects the autonomy of those who have made their data or models openly accessible for research purposes.

Justice. Our research adheres to this principle by ensuring fairness in the design, implementation, and outcomes of our work. We exclusively use publicly available data and open-sourced models, avoiding the arbitrary inclusion or exclusion of any individuals or groups. This approach ensures that no specific population is targeted based on attributes such as religion, socioeconomic status, or technical competency. By using publicly accessible resources, we democratize the benefits of our research, enabling broader access to privacy-preserving techniques for fine-tuning large language models.

Moreover, our focus on memory-efficient methods for differential privacy aligns with the equitable distribution of research burdens and benefits. The computational resources required by current methods often create barriers for researchers and practitioners with limited access to high-performance hardware. By developing approaches that are computationally feasible on local devices, we reduce these barriers, enabling a wider audience to benefit from privacy-preserving technology without imposing undue burdens on any specific group. This equitable approach ensures that the benefits of privacy-preserving fine-tuning are accessible to a diverse and global community, including those from under-resourced environments.

Respect for Law and Public Interest. Our research adheres to this principle by ensuring compliance with applicable legal and ethical standards, as well as promoting transparency and accountability in our methodologies and results. Specifically, for compliance, we exclusively use publicly available datasets and open-sourced models, ensuring that our work avoids the collection, processing, or exposure of sensitive or private information. By utilizing resources that are openly accessible, we mitigate potential legal risks related to privacy, intellectual property, or data ownership. This approach ensures adherence to relevant data protection laws and avoids actions that could be construed as trespassing or unauthorized access to systems. We also prioritize transparency by clearly documenting our methodologies, experimental setups, and results. This allows others to reproduce our findings and assess their implications.

10 Compliance with the Open Science Policy

The artifacts are submitted to for evaluation, and are available at https://zenodo.org/records/15594622. We have prepared the implementation and data necessary to reproduce the main results presented in this paper. The documentation mainly focuses on supporting the key message, that is, our proposed *DP-AggZO* achieves better test accuracy than the previous *DPZero* (which is equivalent to *DP-AggZO* with $K = 1$), and that *DP-AggZO* achieves comparable and sometimes even better utility than *DP-AdamW*, under the same privacy constraints.

References

[1] Martín Abadi, Andy Chu, Ian J. Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and

Li Zhang. Deep learning with differential privacy. In *CCS*, pages 308–318, 2016.

[2] Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *ACL*, pages 7319–7328, 2021.

[3] Kareem Amin, Alex Kulesza, Andrés Muñoz Medina, and Sergei Vassilvitskii. Bounding user contributions: A bias-variance trade-off in differential privacy. In *ICML*, 2019.

[4] Kareem Amin, Alex Kulesza, Andres Munoz, and Sergei Vassilvtiskii. Bounding user contributions: A bias-variance trade-off in differential privacy. In *ICML*, pages 263–271, 2019.

[5] Galen Andrew, Om Thakkar, H. Brendan McMahan, and Swaroop Ramaswamy. Differentially private learning with adaptive clipping. In *NeurIPS*, 2024.

[6] Rohan Anil, Badih Ghazi, Vineet Gupta, Ravi Kumar, and Pasin Manurangsi. Large-scale differentially private BERT. In *EMNLP (Findings)*, pages 6481–6491, December 2022.

[7] Anthropic. The claude 3 model family: Opus, sonnet, haiku, 2024.

[8] Ahmadreza Azizi, Ibrahim Asadullah Tahmid, Asim Waheed, Neal Mangaokar, Jiameng Pu, Mobin Javed, Chandan K. Reddy, and Bimal Viswanath. T-Miner: A generative approach to defend against trojan attacks on DNN-based text classification. In *USENIX Security*, pages 2255–2272, 2021.

[9] Zhongjie Ba, Jieming Zhong, Jiachen Lei, Peng Cheng, Qinglong Wang, Zhan Qin, Zhibo Wang, and Kui Ren. Surrogateprompt: Bypassing the safety filter of text-to-image models via substitution. In *CCS*, page 1166–1180, 2024.

[10] Ashwinkumar Badanidiyuru, Badih Ghazi, Pritish Kamath, Ravi Kumar, Ethan Leeman, Pasin Manurangsi, Avinash V Varadarajan, and Chiyuan Zhang. Optimal unbiased randomizers for regression with label differential privacy. In *NeurIPS*, 2024.

[11] Eugene Bagdasaryan and Vitaly Shmatikov. Blind backdoors in deep learning models. In *USENIX Security*, pages 1505–1521, 2021.

[12] Ergute Bao, Yangfan Jiang, Fei Wei, Xiaokui Xiao, Zitao Li, Yaliang Li, and Bolin Ding. Unlocking the power of differentially private zeroth-order optimization for fine-tuning llms. 2025. URL: https://github.com/erguteb/dp-aggzo/blob/main/tr.pdf.

[13] European Data Protection Board. Report of the work undertaken by the chatgpt taskforce, 2024. URL: https://www.edpb.europa.eu/system/files/2024-05/edpb_20240523_report_chatgpt_taskforce_en.pdf.

[14] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *EMNLP*, pages 632–642, 2015.

[15] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, and et al. Language models are few-shot learners. In *NeurIPS*, pages 1877–1901, 2020.

[16] Zhiqi Bu, Sivakanth Gopi, Janardhan Kulkarni, Yin Tat Lee, Judy Hanwen Shen, and Uthaipon Tantipongpipat. Fast and memory efficient differentially private-sgd via jl projections. In *NeurIPS*, 2024.

[17] Zhiqi Bu, Jialin Mao, and Shiyun Xu. Scalable and efficient training of large convolutional neural networks with differential privacy. In *NeurIPS*, volume 35, pages 38305–38318, 2022.

[18] Zhiqi Bu, Yu-Xiang Wang, Sheng Zha, and George Karypis. Automatic clipping: differentially private deep learning made easier and stronger. In *NeurIPS*, 2024.

[19] Clément L. Canonne, Gautam Kamath, and Thomas Steinke. The discrete gaussian for differential privacy. In *NeurIPS*, 2020.

[20] Nicholas Carlini. Poisoning the unlabeled dataset of semi-supervised learning. In *USENIX Security Symposium*, 2021.

[21] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramèr. Membership inference attacks from first principles. In *SP*, pages 1897–1914, 2022.

[22] Nicholas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramèr, Borja Balle, Daphne Ippolito, and Eric Wallace. Extracting training data from diffusion models. In *USENIX Security*, 2023.

[23] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *USENIX Security*, page 267–284, 2019.

[24] Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom B. Brown, Dawn Xiaodong Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting training data from large language models. In *USENIX Security*, 2020.

[25] Varun Chandrasekaran, Kamalika Chaudhuri, Irene Giacomelli, Somesh Jha, and Songbai Yan. Exploring connections between active learning and model extraction. In *USENIX Security*, 2020.

[26] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, and et al. Evaluating large language models trained on code. *CoRR*, abs/2107.03374, 2021.

[27] Tiejin Chen, Longchao Da, Huixue Zhou, Pingzhi Li, Kaixiong Zhou, Tianlong Chen, and Hua Wei. Privacy-preserving fine-tuning of large language models through flatness. In *SeTLLM @ ICLR*, 2024.

[28] Rishav Chourasia, Jiayuan Ye, and Reza Shokri. Differential privacy dynamics of langevin diffusion and noisy gradient descent. In *NeurIPS*, volume 34, 2021.

[29] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, and et al. Palm: scaling language modeling with pathways. *JMLR*, 24(1), March 2024.

[30] Lynn Chua, Badih Ghazi, Pritish Kamath, Ravi Kumar, Pasin Manurangsi, Amer Sinha, and Chiyuan Zhang. Scalable DP-SGD: Shuffling vs. poisson subsampling. In *NeurIPS*, 2024.

[31] Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognizing textual entailment challenge. In *Proceedings of the PASCAL Challenges Workshop*, pages 177–190, 2005.

[32] Soham De, Leonard Berrada, Jamie Hayes, Samuel L. Smith, and Borja Balle. Unlocking high-accuracy differentially private image classification through scale. In *TPDP at ICML*, 2022.

[33] A. Derumigny and Johannes Schmidt-Hieber. On lower bounds for the bias-variance trade-off. *The Annals of Statistics*, 2020.

[34] Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *PODS*, page 202–210, 2003.

[35] Wei Dong, Juanru Fang, Ke Yi, Yuchao Tao, and Ashwin Machanavajjhala. R2t: Instance-optimal truncation for differentially private query evaluation with foreign keys. In *SIGMOD*, page 759–772, 2022.

[36] Wei Dong, Qiyao Luo, Giulia Fanti, Elaine Shi, and Ke Yi. Almost instance-optimal clipping for summation problems in the shuffle model of differential privacy. In *CCS*, page 1939–1953, 2024.

[37] John C. Duchi, Michael I. Jordan, Martin J. Wainwright, and Andre Wibisono. Optimal rates for zero-order convex optimization: The power of two function evaluations. *TIT*, 61:2788–2806, 2013.

[38] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006.

[39] Václav Fabian. Stochastic approximation methods for constrained and unconstrained systems (harold l. kushner and dean s. clark). *SIAM Review*, 22(3):382–384, 1980. arXiv:https://doi.org/10.1137/1022079, doi:10.1137/1022079.

[40] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Local model poisoning attacks to byzantine-robust federated learning. In *USENIX Security*, 2020.

[41] Vitaly Feldman, Ilya Mironov, Kunal Talwar, and Abhradeep Thakurta. Privacy amplification by iteration. In *FOCS*, pages 521–532, 2018. doi:10.1109/FOCS.2018.00056.

[42] Arun Ganesh, Mahdi Haghifam, Thomas Steinke, and Abhradeep Guha Thakurta. Faster differentially private convex optimization via second-order methods. In *NeurIPS*, 2023.

[43] Google Gemini Team. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[44] Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. An investigation into neural net optimization via hessian eigenvalue density. In *ICML*, pages 2232–2241, 2019.

[45] Eric Goldman. An introduction to the california consumer privacy act (ccpa). *Santa Clara Univ. Legal Studies Research Paper*, 2020.

[46] Aaron Grattafiori and Meta AI. The llama 3 herd of models. *arXiv*, 2024.

[47] Jingxuan He and Martin Vechev. Large language models for code: Security hardening and adversarial testing. In *CCS*, page 1865–1879, 2023.

[48] Xinlei He, Xinyue Shen, Zeyuan Chen, Michael Backes, and Yang Zhang. Mgtbench: Benchmarking machine-generated text detection. In *CCS*, page 2251–2265, 2024.

[49] Chris Jay Hoofnagle, Bart Van Der Sloot, and Frederik Zuiderveen Borgesius. The european union general data protection regulation: what it is and what it means. *Information & Communications Technology Law*, 28(1):65–98, 2019.

[50] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press Series in Applied Mathematics and Computational Science. Cambridge University Press, 1990.

[51] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *ICLR*, 2022.

[52] Ziyue Huang, Yuting Liang, and Ke Yi. Instance-optimal mean estimation under differential privacy. In *NeurIPS*, 2021.

[53] Hengrui Jia, Christopher A. Choquette-Choo, Varun Chandrasekaran, and Nicolas Papernot. Entangled watermarks as a defense against model extraction. In *USENIX Security*, pages 1937–1954, 2021.

[54] Gautam Kamath, Argyris Mouzakis, Matthew Regehr, Vikrant Singhal, Thomas Steinke, and Jonathan Ullman. A bias-accuracy-privacy trilemma for statistical estimation. *JASA*, pages 1–23, 12 2024.

[55] Kavita Kumari, Alessandro Pegoraro, Hossein Fereidooni, and Ahmad-Reza Sadeghi. Demasq: Unmasking the chatgpt wordsmith. 2023.

[56] Xabier Lareo. TechSonar report: Large language models (LLM), 2024. URL: https://www.edps.europa.eu/data-protection/technology-monitoring/techsonar/large-language-models-llm_en.

[57] Klas Leino and Matt Fredrikson. Stolen memories: leveraging model memorization for calibrated white-box membership inference. In *USENIX Security*, 2020.

[58] Tian Li, Manzil Zaheer, Ken Liu, Sashank J. Reddi, Hugh Brendan McMahan, and Virginia Smith. Differentially private adaptive optimization with delayed preconditioners. In *ICLR*, 2023.

[59] Xuechen Li, Florian Tramer, Percy Liang, and Tatsunori Hashimoto. Large language models can be strong differentially private learners. In *International Conference on Learning Representations*, 2022.

[60] Xiyang Liu, Prateek Jain, Weihao Kong, Sewoong Oh, and Arun Sai Suggala. Label robust and differentially private linear regression: Computational and statistical efficiency. In *NeurIPS*, 2023.

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

[62] Yiyong Liu, Zhengyu Zhao, Michael Backes, and Yang Zhang. Membership inference attacks by exploiting loss trajectory. In *CCS*, page 2085–2098, 2022.

[63] Z Liu, J Lou, W Bao, Y Hu, B Li, Z Qin, and K Ren. Differentially private zeroth-order methods for scalable large language model finetuning. In *NDSS*, 2025.

[64] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

[65] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

[66] Nils Lukas, Ahmed Salem, Robert Sim, Shruti Tople, Lukas Wutschitz, and Santiago Zanella-Beguelin. Analyzing Leakage of Personally Identifiable Information in Language Models . In *SP*, pages 346–363, 2023.

[67] Peizhuo Lv, Chang Yue, Ruigang Liang, Yunfei Yang, Shengzhi Zhang, Hualong Ma, and Kai Chen. A data-free backdoor injection approach in neural networks. In *USENIX Security*, 2023.

[68] Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D. Lee, Danqi Chen, and Sanjeev Arora. Fine-tuning language models with just forward passes. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2024. Curran Associates Inc.

[69] Charles H. Martin and Michael W. Mahoney. Implicit self-regularization in deep neural networks: evidence from random matrix theory and implications for learning. *JMLR*, 22(1), January 2021.

[70] Ilya Mironov. Rényi differential privacy. In *CSF*, pages 263–275, 2017.

[71] Ilya Mironov, Kunal Talwar, and Li Zhang. Rényi differential privacy of the sampled gaussian mechanism. *CoRR*, abs/1908.10530, 2019.

[72] Milad Nasr, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A. Feder Cooper, Daphne Ippolito, Christopher A. Choquette-Choo, Eric Wallace, Florian Tramèr, and Katherine Lee. Scalable extraction of training data from (production) language models, 2023. arXiv:2311.17035.

[73] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Standalone and federated learning under passive and active white-box inference attacks. In *S&P*, pages 739–753, 2019.

[74] Thien Duc Nguyen, Phillip Rieger, Huili Chen, Hossein Yalame, Helen Möllering, Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Shaza Zeitouni, Farinaz Koushanfar, Ahmad-Reza Sadeghi, and Thomas Schneider. FLAME: Taming backdoors in federated learning. In *USENIX Security*, pages 1415–1432, 2022.

[75] OpenAI. Gpt-4 technical report. *arXiv*, 2023.

[76] Ren Pang, Zhaohan Xi, Shouling Ji, Xiapu Luo, and Ting Wang. On the security risks of AutoML. In *USENIX Security*, pages 3953–3970, 2022.

[77] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. Scalable private learning with PATE. In *ICLR*, 2018.

[78] Vardan Papyan. Traces of class/cross-class structure pervade deep learning spectra. *Journal of Machine Learning Research*, 21(252):1–64, 2020.

[79] Venkatadheeraj Pichapati, Ananda Theertha Suresh, Felix X. Yu, Sashank J. Reddi, and Sanjiv Kumar. Adaclip: Adaptive clipping for private sgd, 2019. arXiv:1908.07643.

[80] Jiameng Pu, Zain Sarwar, Sifat Muhammad Abdullah, Abdullah Rehman, Yoonjin Kim, Parantapa Bhattacharya, Mobin Javed, and Bimal Viswanath. Deepfake Text Detection: Limitations and Opportunities . In *SP*, pages 1613–1630, May 2023.

[81] Zhen Qin, Daoyuan Chen, Bingchen Qian, Bolin Ding, Yaliang Li, and Shuiguang Deng. Federated full-parameter tuning of billion-sized language models with communication cost under 18 kilobytes. In *ICML*, 2024.

[82] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2383–2392, 2016.

[83] Matthew Regehr. *A Bias-Variance-Privacy Trilemma for Statistical Estimation*. PhD thesis, University of Waterloo, Waterloo, Ontario, Canada, 2023. Master of Mathematics in Computer Science.

[84] Alfréd Rényi. On measures of entropy and information. In *Berkeley symposium on mathematical statistics and probability*, volume 1, pages 547–562, 1961.

[85] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951. doi:10.1214/aoms/1177729586.

[86] Roei Schuster, Congzheng Song, Eran Tromer, and Vitaly Shmatikov. You autocomplete me: Poisoning vulnerabilities in neural code completion. In *USENIX Security*, pages 1559–1575, 2021.

[87] Shawn Shan, Arjun Nitin Bhagoji, Haitao Zheng, and Ben Y. Zhao. Poison forensics: Traceback of data poisoning attacks in neural networks. In *USENIX Security*, pages 3575–3592, 2022.

[88] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *S&P*, pages 3–18, 2017.

[89] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, pages 1631–1642, October 2013.

[90] Congzheng Song, Thomas Ristenpart, and Vitaly Shmatikov. Machine learning models that remember too much. In *CCS*, page 587–601, 2017.

[91] Liwei Song and Prateek Mittal. Systematic evaluation of privacy risks of machine learning models. In *USENIX Security*, 2020.

[92] J.C. Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, 37(3):332–341, 1992. doi:10.1109/9.119632.

[93] Max Staats, Matthias Thamm, and Bernd Rosenow. Small singular values matter: A random matrix analysis of transformer models, 2025. arXiv:2410.17770.

[94] Kunal Talwar, Shan Wang, Audra McMillan, Vojta Jina, Vitaly Feldman, Bailey Basile, Aine Cahill, Yi Sheng Chan, Mike Chatzidakis, Junye Chen, et al. Samplable anonymous aggregation for private federated data analysis. *arXiv preprint arXiv:2307.15017*, 2023.

[95] Kunsheng Tang, Wenbo Zhou, Jie Zhang, Aishan Liu, Gelei Deng, Shuai Li, Peigui Qi, Weiming Zhang, Tianwei Zhang, and NengHai Yu. Gendercare: A comprehensive framework for assessing and reducing gender bias in large language models. In *CCS*, page 1196–1210, 2024.

[96] Xinyu Tang, Ashwinee Panda, Milad Nasr, Saeed Mahloujifar, and Prateek Mittal. Private fine-tuning of large language models with zeroth-order optimization, 2024. arXiv:2401.04343.

[97] Florian Tramèr, Gautam Kamath, and Nicholas Carlini. Position: Considerations for differentially private learning with large-scale public pretraining. In *ICML*, 2024.

[98] Florian Tramèr, Gautam Kamath, and Nicholas Carlini. Position: Considerations for differentially private learning with large-scale public pretraining. In *ICML*, pages 48453–48467, 2024.

[99] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *USENIX Security*, page 601–618, 2016.

[100] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, page 6000–6010, 2017.

[101] Ellen M. Voorhees. Overview of the trec 2000 question answering track. In *Proceedings of the 9th Text REtrieval Conference (TREC-9)*, pages 42–51, 2000.

[102] Boxin Wang, Fan Wu, Yunhui Long, Luka Rimanic, Ce Zhang, and Bo Li. Datalens: Scalable privacy preserving training via gradient compression and aggregation. In *CCS*, page 2146–2168, 2021.

[103] Haodi Wang, Kai Dong, Zhilei Zhu, Haotong Qin, Aishan Liu, Xiaolin Fang, Jiakai Wang, and Xianglong Liu. Transferable multimodal attack on vision-language pretraining models. In *SP*, pages 1722–1740, 2024.

[104] Adina Williams, Nikita Nangia, and Samuel R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 1112–1122. Association for Computational Linguistics, 2018.

[105] Hanshen Xiao, Jun Wan, and Srinivas Devadas. Geometry of sensitivity: Twice sampling and hybrid clipping in differential privacy with optimal gaussian noise and application to deep learning. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, CCS '23, page 2636–2650, 2023.

[106] Hanshen Xiao, Zihang Xiang, Di Wang, and Srinivas Devadas. A theory to instruct differentially-private learning via clipping bias reduction. In *S&P*, pages 2170–2189, 2023.

[107] Yuchen Yang, Bo Hui, Haolin Yuan, Neil Gong, and Yinzhi Cao. Sneakyprompt: Jailbreaking text-to-image generative models. In *SP*, pages 897–912, 2024.

[108] Zhewei Yao, Amir Gholami, Kurt Keutzer, and Michael W. Mahoney. Pyhessian: Neural networks through the lens of the hessian. *Big Data*, pages 581–590, 2019.

[109] Jiayuan Ye, Aadyaa Maddi, Sasi Kumar Murakonda, Vincent Bindschaedler, and Reza Shokri. Enhanced membership inference attacks against machine learning models. In *CCS*, page 3093–3106, 2022.

[110] Da Yu, Saurabh Naik, Arturs Backurs, Sivakanth Gopi, Huseyin A. Inan, Gautam Kamath, Janardhan Kulkarni, Yin Tat Lee, Andre Manoel, Lukas Wutschitz, Sergey Yekhanin, and Huishuai Zhang. Differentially private fine-tuning of language models. *JPC*, 14(2), 2024.

[111] Santiago Zanella-Béguelin, Lukas Wutschitz, Shruti Tople, Victor Rühle, Andrew Paverd, Olga Ohrimenko, Boris Köpf, and Marc Brockschmidt. Analyzing information leakage of updates to natural language models. In *CCS*, page 363–375, 2020.

[112] Jiawei Zhang, Zhongzhu Chen, Huan Zhang, Chaowei Xiao, and Bo Li. DiffSmooth: Certifiably robust learning via diffusion models and local smoothing. In *USENIX Security*, pages 4787–4804, 2023.

[113] Liang Zhang, Bingcong Li, Kiran Koshy Thekumparampil, Sewoong Oh, and Niao He. Dpzero: Private fine-tuning of language models without backpropagation, 2024. arXiv:2310.09639.

[114] Qinzi Zhang, Hoang Tran, and Ashok Cutkosky. Private zeroth-order nonsmooth nonconvex optimization. In *ICLR*, 2024.

[115] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona T. Diab, and et al. OPT: open pre-trained transformer language models. *CoRR*, abs/2205.01068, 2022.

[116] Xinwei Zhang, Zhiqi Bu, Steven Wu, and Mingyi Hong. Differentially private SGD without clipping bias: An error-feedback approach. In *ICLR*, 2024.

[117] Zhuo Zhang, Guanhong Tao, Guangyu Shen, Shengwei An, Qiuling Xu, Yingqi Liu, Yapeng Ye, Yaoxuan Wu, and Xiangyu Zhang. PELICAN: Exploiting backdoors of naturally trained deep learning models in binary code analysis. In *USENIX Security*, pages 2365–2382, 2023.

[118] Yuqing Zhu and Yu-Xiang Wang. Poission subsampled Rényi differential privacy. In *ICML*, pages 7634–7642, 2019.

Algorithm 2: DPZero [113]

Input: private dataset D of size n; total number of training steps T; subsampling rate q; learning rate η; initial (pre-trained) model parameters $\boldsymbol{\theta}_0$; perturbation scale ϕ; Gaussian noise multiplier σ_m; clipping threshold c; Laplace noise parameter ψ for perturbing n.

Perturb the size of $\mathcal{D}$: $\tilde{n} = n + \text{Lap}(0, \psi)$.
Compute targeted batch size: $b = q\tilde{n}$.
for $t = 1 \ldots T$ **do**
  Obtain batch $\mathcal{B}_t$ using Poisson sampling, where each record in $\mathcal{D}$ is sampled with probability q .
  Sample $\mathbf{z}$ from the standard Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{I}_d)$. // d is the dimension of model parameters
  Compute $\boldsymbol{\theta}_t^+ \leftarrow \boldsymbol{\theta}_t + \phi \cdot \mathbf{z}_k$.
  Compute $\boldsymbol{\theta}_t^+ \leftarrow \boldsymbol{\theta}_t - \phi \cdot \mathbf{z}_k$.
  for $x \in \mathcal{B}_t$ **do**
    Query the model: $\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x) = \frac{\mathcal{L}(\boldsymbol{\theta}_t^+; x) - \mathcal{L}(\boldsymbol{\theta}_t^-; x)}{2\phi}$
    Artificial clipping: $\widehat{\Delta}_{\mathbf{z}}(\boldsymbol{\theta}_t; x) = \frac{\min(|\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)|, c)}{c} \cdot \Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)$.
  Noise injection: $\widetilde{\Delta}_{\mathbf{z}}(\boldsymbol{\theta}_t; \mathcal{B}) = \sum_{x \in B} \widehat{\Delta}_{\mathbf{z}}(\boldsymbol{\theta}_t; x) + \mathcal{N}(0, \sigma_m^2 c^2)$.
  Update model parameters $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \frac{\eta}{b} \cdot \widetilde{\Delta}_{\mathbf{z}}(\boldsymbol{\theta}_t; \mathcal{B}) \cdot \mathbf{z}$.

Output: Model parameters θ.

A Appendix

A.1 Additional Background

Running the Gaussian mechanism on a random subset of the input also satisfies RDP [71, 118].

Lemma 6 (Subsampled Gaussian mechanism [71, 118])**.** *Let $\mathcal{M}$ be a mechanism that satisfies $(h, \varepsilon(h))$-RDP for $h = 2, \ldots, \alpha$ ($\alpha \in \mathbb{Z}, \alpha \geq 2$) by injecting Gaussian noises, and S_q be a procedure that uniformly samples each record of the input data with probability q (i.e., Poisson sampling). Then $\mathcal{M} \circ S_q$ satisfies (α, ε)-RDP with*

$$\varepsilon = \frac{1}{\alpha - 1} \cdot \log \Bigg((1-q)^{\alpha-1} (\alpha q - q + 1) + \sum_{h=2}^{\alpha} \binom{\alpha}{h} (1-q)^{\alpha-h} q^h e^{(h-1)\varepsilon(h)} \Bigg).$$

In particular, when $\mathcal{M}$ injects noise sampled from $\mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_d)$ to the outcome of function F of sensitivity c, then we have $\varepsilon(h) = \frac{hc^2}{2\sigma^2}$.

The composition of RDP mechanisms also satisfies RDP.

Lemma 7 (Composition of RDP Mechanisms [70])**.** *If mechanisms $\mathcal{M}_1, \ldots, \mathcal{M}_T$ satisfy $(\alpha, \varepsilon_1), \ldots, (\alpha, \varepsilon_T)$-RDP, respectively, then, $\mathcal{M}_1 \circ \ldots \circ \mathcal{M}_T$ satisfies $(\alpha, \sum_{t=1}^T \varepsilon_t)$-RDP.*

Finally, we can convert the RDP guarantee of a mechanism to the classic (ε, δ)-DP one [19].

Lemma 8 (Conversion from RDP to (ε, δ)-DP [19])**.** *If mechanism $\mathcal{M}$ satisfies $(\alpha, \varepsilon(\alpha))$-RDP for some $\alpha \in (1, \infty)$, then $\mathcal{M}$ satisfies (ε, δ)-DP for*

$$\varepsilon = \varepsilon(\alpha) + \frac{\log(1/\delta) + (\alpha - 1)\log(1 - 1/\alpha) - \log(\alpha)}{\alpha - 1}.$$

DPSGD. After obtaining the noisy gradient sum G_{dpsgd} as in equation 5. The model is updated to $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \cdot \frac{G_{\text{dpsgd}}}{b}$, where b is the targeted batch size. When using Poisson sampling (see e.g., [30, 71]), the targeted batch size is set to $q\tilde{n}$ where $\tilde{n}$ stands for the DP estimate for n (the perturbation for n typically takes 5% of the overall privacy budget). The overall privacy cost (i.e., the privacy parameters) for running *DPSGD* for T iterations/updates is the sum of costs for releasing $\tilde{n}$ for once and releasing $\{G_{\text{dpsgd}}(\boldsymbol{\theta}_t; \mathcal{B}_t)\}_{t=1}^T$.

DPZero. We list the complete procedure of *DPZero* as in Algorithm 2.

A.2 Experiment Details

We provide the memory consumption on OPT models in Table 5. We provide the search grid for the hyperparameters of DP methods in Tables 6, 7, and 8. For non-DP methods, the hyperparameters are the same except that there is no artificial clipping. The results of *DP-AdamW* on model RoBERTa (355M) are taken from [113]. The hyperparameters used for *DP-AdamW* on OPT-1.3B are in Table 8 (run out of memory for OPT-6.7B). For OPT models, the size of the training data in SST-2 and SQuAD is 1000. For RoBERTa (355M), the size of the training data vary on different datasets, as shown in Table 9.

Comparison with [63]. *DP-ZOPO* [63] applies pruning over the model parameters (i.e., only a subset of the parameters are updated in each iteration) with a stage-wise selection for the hyperparameters (i.e., in different iterations, the hyperparameters could be different, e.g., the perturbation scale parameter ϕ). Notably, such modifications have no impact on the privacy cost, as they are either fixed before computing the model's losses on the private input data or are determined according to the updated model parameters (post-processing preserves DP). *DP-ZOPO* achieves higher utility than *DPZero*. However, as evidenced in their paper [61], the performance gap between *DP-AdamW* is still notable. Our *DP-AggZO*, by reducing the clipping error due to DP, achieves much higher performance compared with *DP-ZOPO* while surpassing *DPSGD* in several benchmark tasks. E.g., *DP-AggZO* on MNLI using RoBERTa (355M) achieves an accuracy of 78.2% under the privacy constraint of $\varepsilon = 6, \delta = 10^{-5}$ whereas *DP-ZOPO* achieves 74.8% using the same model but under a less restrictive privacy constraint of $\varepsilon = 8, \delta \approx 6 \times 10^{-4}$.

Table 8: Search grid for the hyperparameters of *DP-AdamW*.

Hyperparameter	Value
Number of epochs	$\{1,5,10,20,25\}$
Subsampling rate	Rates that lead to expected batch size of $\{8,32,64\}$
Clipping threshold	$1,10,100,200$
Learning rate	$10^{-5},5\times10^{-5},10^{-4}$

Table 9: Number of training records for each datasets used for RoBERTa (355M) and OPT models and the corresponding subsampling rate for Poisson sampling if we want the expected batch size to be 8. We keep five decimal points.

Dataset	Training records	Subampling rate
For RoBERTa (355M)		
SST-2	1024	0.00781
SST-5	2560	0.00313
SNLI	1536	0.00521
MNLI	1536	0.00521
RTE	1024	0.00781
TREC	2646	0.00302
For OPT-1.3B & OPT-6.7B		
SST-2	1000	0.008
SQuAD	1000	0.008

Table 5: Peak memory usage of DP-AdamW, DPZero, and DP-AggZO (with $K=16,64$, and 256), tested on a H20 GPU and OPT models. We measure the memory usage in GiB, with OOM standing for out of memory. **The usage of DP-AggZO with different K's is identical.**

Model	Dataset	Algorithm	Memory
OPT-1.3B	SST-2	DP-AdamW	**31.25** GiB
		DPZero	4.13 GiB
		DP-AggZO	4.13 GiB
OPT-1.3B	SQuAD	DP-AdamW	**35.4** GiB
		DPZero	7.03 GiB
		DP-AggZO	7.03 GiB
OPT-6.7B	SST-2	DP-AdamW	**OOM**
		DPZero	15.24 GiB
		DP-AggZO	15.24 GiB
OPT-6.7B	SQuAD	DP-AdamW	**OOM**
		DPZero	18.81 GiB
		DP-AggZO	18.81 GiB

Table 6: Search grid for the hyperparameters of *DP-AggZO*.

Hyperparameter	Value
Number of iterations	1000
Subsampling rate	Rates that lead to expected batch size of $\{8,32,64\}$
Clipping threshold	$\{1,1.25\}\times$ $\{1,2,3,4,6,8,10,20,40\}$
Learning rate	$\{1,2,4,8\}\times\{10^{-6},10^{-5}\}$

Table 7: Search grid for the hyperparameters of *DPZero*.

Hyperparameter	Value
Number of iterations	$\{1,2,5,10,20,50,100\}\times10^3$
Subsampling rate	Rates that lead to expected batch size of $\{8,32,64\}$
Clipping threshold	$\{100,200,400\}$
Learning rate	$\{1,5\}\times\{10^{-7},10^{-6},10^{-5}\}$

A.3 Analyses

Our analyses are based on the assumption that the step size $\phi\to0$, and that the models we are fine-tuning have low effective ranks (see Assumption 1 and Assumption 2).

We first present a proposition that is useful to our analyses. Recall that $\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t;x)\cdot\mathbf{z}$ stands for the "approximate" zeroth-order gradient. We are interested in the maximum eigenvalue for the matrix

$$\begin{aligned}\boldsymbol{\Sigma}=&\operatorname{cov}(\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t;x)\cdot\mathbf{z})\\=&\mathbb{E}\left[\left(\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t;x)\cdot\mathbf{z}\right)\left(\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t;x)\cdot\mathbf{z}\right)^T\right]\\&-\mathbb{E}\left[\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t;x)\cdot\mathbf{z}\right]\mathbb{E}\left[\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t;x)\cdot\mathbf{z}\right]^T.\end{aligned}$$

It is easy to verify that $\boldsymbol{\Sigma}$ is positive semi-definite and all its eigenvalues are positive. Next, we establish an upper bound for $\lambda_{\max}(\boldsymbol{\Sigma})$, the maximum eigenvalue of $\boldsymbol{\Sigma}$.

Proposition 2. *When $\phi\to0$, the maximum eigenvalue of $\boldsymbol{\Sigma}$ satisfies*

$$\lambda_{\max}(\boldsymbol{\Sigma})\le3\cdot\mathbb{E}[\|\nabla\mathcal{L}(\boldsymbol{\theta}_t;x)\|^2]. \tag{31}$$

Proof. When $\phi\to0$, we can show that $\frac{\mathcal{L}(\theta_t+\phi\mathbf{z};x)-\mathcal{L}(\theta_t-\phi\mathbf{z};x)}{2\phi}=\mathbf{z}^T\nabla\mathcal{L}(\theta_t;x)$. Hence, we have

$$\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t;x)=\mathbf{z}^T\nabla\mathcal{L}. \tag{32}$$

Since $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, we have that $\mathbb{E}[\mathbf{z}\mathbf{z}^T] = \mathbf{I}$. Hence, $\mathbb{E}[\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x) \cdot \mathbf{z}] = \nabla \mathcal{L}(\theta_t; x)$. Then we have

$$\mathbb{E}\left[\left(\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x) \cdot \mathbf{z}\right)\left(\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x) \cdot \mathbf{z}\right)^T\right] = \mathbb{E}[(\mathbf{z}^T \nabla \mathcal{L}(\theta_t; x))^2 \mathbf{z}\mathbf{z}^T]$$

For each $i, j \in [d]$, we have

$$\begin{aligned}
&\left[\mathbb{E}\left[\left(\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x) \cdot \mathbf{z}\right)\left(\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x) \cdot \mathbf{z}\right)^T\right]\right]_{ij} \\
&= \mathbb{E}[(\mathbf{z}^T \nabla \mathcal{L}(\boldsymbol{\theta}_t; x))^2 z_i z_j] \\
&= \mathbb{E}[(\sum_{r,s \in [d]} \nabla \mathcal{L}(\boldsymbol{\theta}_t; x)_r \nabla \mathcal{L}(\boldsymbol{\theta}_t; x)_s) z_r z_s z_i z_j]
\end{aligned}$$

Since $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, we have $\mathbb{E}[z_r z_s z_i z_j] = \delta_{rs}\delta_{ij} + \delta_{ri}\delta_{sj} + \delta_{rj}\delta_{si}$. Here we have adopted the notation of Kronecker delta: $\delta_{ij} = 1$ if $i = j$; and $\delta_{ij} = 0$ if $i \neq j$.

$$\begin{aligned}
\boldsymbol{\Sigma} =& \mathbb{E}[\|\nabla \mathcal{L}(\boldsymbol{\theta}_t; x)\|^2] \cdot \mathbf{I}_d + 2 \cdot \mathbb{E}[\nabla \mathcal{L}(\boldsymbol{\theta}_t; x) \nabla \mathcal{L}(\boldsymbol{\theta}_t; x)^T] \\
&- \mathbb{E}[\nabla \mathcal{L}(\boldsymbol{\theta}_t; x)] \mathbb{E}[\nabla \mathcal{L}(\boldsymbol{\theta}_t; x)]^T \\
=& \mathbb{E}[\|\nabla \mathcal{L}(\boldsymbol{\theta}_t; x)\|^2] \cdot \mathbf{I}_d + 2 \cdot \mathrm{cov}(\nabla \mathcal{L}(\boldsymbol{\theta}_t; x)) \\
&+ \mathbb{E}[\nabla \mathcal{L}(\boldsymbol{\theta}_t; x)] \mathbb{E}[\nabla \mathcal{L}(\boldsymbol{\theta}_t; x)]^T
\end{aligned}$$

Here we adopt the Weyl's inequality [50] and get

$$\begin{aligned}
\lambda_{\max}(\boldsymbol{\Sigma}) \leq& \lambda_{\max}(\mathbb{E}[\|\nabla \mathcal{L}(\boldsymbol{\theta}_t; x)\|^2] \cdot \mathbf{I}_d) \\
&+ 2 \cdot \lambda_{\max}(\mathrm{cov}(\nabla \mathcal{L}(\boldsymbol{\theta}_t; x))) \\
&+ \lambda_{\max}(\mathbb{E}[\nabla \mathcal{L}(\boldsymbol{\theta}_t; x)] \mathbb{E}[\nabla \mathcal{L}(\boldsymbol{\theta}_t; x)]^T) \\
=& \mathbb{E}[\|\nabla \mathcal{L}(\boldsymbol{\theta}_t; x)\|^2] \\
&+ 2 \cdot (\mathbb{E}[\|\nabla \mathcal{L}(\boldsymbol{\theta}_t; x)\|^2] - \|\mathbb{E}[\nabla \mathcal{L}(\boldsymbol{\theta}_t; x)]\|^2) \\
&+ \|\mathbb{E}[\nabla \mathcal{L}(\boldsymbol{\theta}_t; x)]\|^2 \\
=& 3 \cdot \mathbb{E}[\|\nabla \mathcal{L}(\boldsymbol{\theta}_t; x)\|^2] - \|\mathbb{E}[\nabla \mathcal{L}(\boldsymbol{\theta}_t; x)]\|^2 \\
\leq& 3 \cdot \mathbb{E}[\|\nabla \mathcal{L}(\boldsymbol{\theta}_t; x)\|^2]
\end{aligned}$$

Alternatively, we can write

$$\mathrm{tr}(\mathrm{cov}(\nabla \mathcal{L}(\boldsymbol{\theta}_t; x)) = \mathbb{E}[\|\nabla \mathcal{L}(\boldsymbol{\theta}_t; x)\|^2] - \|\nabla \mathcal{L}(\boldsymbol{\theta}_t)\|^2.$$

Then we have

$$\lambda_{\max}(\boldsymbol{\Sigma}) \leq 3\mathrm{tr}(\mathrm{cov}(\nabla \mathcal{L}(\boldsymbol{\theta}_t; x)) + 2\|\nabla \mathcal{L}(\boldsymbol{\theta}_t)\|^2.$$

□

Next, we prove the convergence for DPZO. We first recall a standard assumption for obtaining a dimension-free the convergence rate (i.e., independent of the model dimension d) of fine-tuning large language models is the assumption of low effective rank [44, 78, 108]. Such assumption was also adopted in prior work for analyzing the convergence of zeroth-order methods [68, 96, 113] and serves as the theoretical foundation for PEFT methods [2, 51].

Assumption 1 (Local r-effective rank). *Let $G(\boldsymbol{\theta}_t) = \max_{x \in \mathcal{D}} \|\nabla \mathcal{L}(\boldsymbol{\theta}_t; x)\|$. There exists a matrix $\mathbf{H}(\boldsymbol{\theta}_t) \preceq l \cdot \mathbf{I}_d$ (where d is the model dimension) such that:*

1. For all $\boldsymbol{\theta}$ such that $\|\boldsymbol{\theta} - \boldsymbol{\theta}_t\| \leq \eta d G(\boldsymbol{\theta}_t)$, we have $\nabla^2 \mathcal{L}(\boldsymbol{\theta}) \preceq \mathbf{H}(\boldsymbol{\theta}_t)$.

2. The effective rank of $\mathbf{H}(\boldsymbol{\theta}_t)$ is at most r, i.e., $tr(\mathbf{H}(\boldsymbol{\theta}_t)/\|\mathbf{H}(\boldsymbol{\theta}_t)\|_{op} \leq r$.

In the context of DP, to adapt to the injection of random noises, which may make the distance $\|\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t\|$ unbounded, the constraints for the above inequalities to hold are often relaxed as follows (e.g., see Assumption 3.5 in [113]).

Assumption 2 (Global r-effective rank). *There exists a matrix $\mathbf{H} \preceq l \cdot \mathbf{I}_d$ (where d is the model dimension) such that:*

1. For all $\boldsymbol{\theta}$, we have $\nabla^2 \mathcal{L}(\boldsymbol{\theta}) \preceq \mathbf{H}$.

2. $tr(\mathbf{H}(\boldsymbol{\theta}_t)/\|\mathbf{H}(\boldsymbol{\theta}_t)\|_{op} \leq r$.

Proof of Lemma 2. By the integral form of Taylor's Theorem,

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}_{t+1}) =& \mathcal{L}(\boldsymbol{\theta}_t) + \nabla \mathcal{L}(\boldsymbol{\theta}_t)^T (\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t) \\
&+ \int_0^1 \gamma (\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t)^T \nabla^2 \mathcal{L}(\boldsymbol{\theta}_t + \gamma(\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t))(\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t) d\gamma.
\end{aligned}$$

Under Assumption 2, we plug in the update rule for $\boldsymbol{\theta}_{t+1}$, i.e., $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_{t-1} - \eta \frac{\left(\sum_{x \in \mathcal{B}_t} \widehat{\Delta}_{\mathbf{z}}(\boldsymbol{\theta}_t; x)\right) + u}{b} \cdot \mathbf{z}$, where b is the targeted batch size, and get

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}_{t+1}) \leq& \mathcal{L}(\boldsymbol{\theta}_t) + \nabla \mathcal{L}(\boldsymbol{\theta}_t)^T (\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t) \\
&+ \frac{1}{2} (\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t)^T \mathbf{H} (\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t) \\
=& \mathcal{L}(\boldsymbol{\theta}_t) - \frac{\eta}{b} \cdot \nabla \mathcal{L}(\boldsymbol{\theta}_t)^T \left(\left(\sum_{x \in \mathcal{B}} \widehat{\Delta}_{\mathbf{z}}(\boldsymbol{\theta}_t; x) \right) + u \right) \cdot \mathbf{z} \\
&+ \frac{\eta^2 \left(\left(\sum_{x \in \mathcal{B}} \widehat{\Delta}_{\mathbf{z}}(\boldsymbol{\theta}_t; x) \right) + u \right)^2}{2b^2} \cdot \mathbf{z}^T \mathbf{H} \mathbf{z}.
\end{aligned}$$

We then take conditional expectation with respect to $\boldsymbol{\theta}_t$ on both sides, and get

$$\begin{aligned}
&\mathbb{E}[\mathcal{L}(\boldsymbol{\theta}_{t+1}) | \boldsymbol{\theta}_t] - \mathcal{L}(\boldsymbol{\theta}_t) \\
\leq& -\frac{\eta}{b} \cdot \mathbb{E}\left[\nabla \mathcal{L}(\boldsymbol{\theta}_t)^T \left(\left(\sum_{x \in \mathcal{B}} \hat{\Delta}_{\mathbf{z}}(\boldsymbol{\theta}_t; x) \right) + u \right) \cdot \mathbf{z} \right] \\
&+ \frac{\eta^2 \cdot |\mathcal{B}_t|^2 \cdot (1 + \sigma_m^2) \cdot l \cdot r \cdot c^2}{2b^2} \\
=& -\frac{\eta}{b} \cdot \nabla \mathcal{L}(\boldsymbol{\theta}_t)^T \mathbb{E}\left[\sum_{x \in \mathcal{B}_t} \hat{\Delta}_{\mathbf{z}}(\boldsymbol{\theta}_t; x) \cdot \mathbf{z} \right] \\
&+ \frac{\eta^2 \cdot |\mathcal{B}_t|^2 \cdot (1 + \sigma_m^2) \cdot l \cdot r \cdot c^2}{2b^2}
\end{aligned}$$

In the first inequality, we have utilized the fact that u is of zero mean, u and $\mathbf{z}$ are independent, each $\hat{\Delta}_{\mathbf{z}}(\boldsymbol{\theta}_t; x)$ is in the

range of $[-c,+c]$, and that $\mathbb{E}[\mathbf{z}^T\mathbf{H}\mathbf{z}] = tr(\mathbf{H}) \leq l \cdot r$ when $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$. In the second equality, we have got rid of the term $-\frac{\eta}{b} \cdot \mathbb{E}\Big[\nabla \mathcal{L}(\boldsymbol{\theta}_t)^T u \cdot \mathbf{z}\Big]$ since u is of zero mean, u and $\mathbf{z}$ are independent. The sum over records in batch $\mathcal{B}_t$ disappears due to the linearity of expectation. Next, we plug in the relationship between $\sum_{x\in\mathcal{B}_t} \hat{\Delta}_{\mathbf{z}}(\boldsymbol{\theta}_t; x)$ and $\sum_{x\in\mathcal{B}_t} \Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)$ and get

$$
\begin{aligned}
&\mathbb{E}[\mathcal{L}(\boldsymbol{\theta}_{t+1})|\boldsymbol{\theta}_t] - \mathcal{L}(\boldsymbol{\theta}_t) \\
\leq& -\frac{\eta}{b} \cdot \nabla \mathcal{L}(\boldsymbol{\theta}_t)^T \mathbb{E}\Big[\Big((1 - err(\boldsymbol{\theta}, \mathcal{B}_t, \mathbf{z})\Big) \cdot \sum_{x\in\mathcal{B}_t} \Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x) \cdot \mathbf{z}\Big] \\
&+ \frac{\eta^2 \cdot |\mathcal{B}_t|^2 \cdot (1+\sigma_m^2) \cdot l \cdot r \cdot c^2}{2b^2} \\
=& -\frac{\eta|\mathcal{B}_t|}{b} \cdot \|\nabla \mathcal{L}(\boldsymbol{\theta}_t)\|^2 + \frac{\eta^2 \cdot |\mathcal{B}_t|^2 \cdot (1+\sigma_m^2) \cdot l \cdot r \cdot c^2}{2b^2} \\
&+ \frac{\eta}{b} \cdot \nabla \mathcal{L}(\boldsymbol{\theta}_t)^T \mathbb{E}\Big[err(\boldsymbol{\theta}, \mathcal{B}_t, \mathbf{z}) \sum_{x\in\mathcal{B}_t} \Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x) \cdot \mathbf{z}\Big]
\end{aligned}
$$

We work on the last term.

$$
\begin{aligned}
&\nabla \mathcal{L}(\boldsymbol{\theta}_t)^T \mathbb{E}\Big[err(\boldsymbol{\theta}, \mathcal{B}_t, \mathbf{z}) \sum_{x\in\mathcal{B}_t} \Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x) \cdot \mathbf{z}\Big] \\
=& \nabla \mathcal{L}(\boldsymbol{\theta}_t)^T \Bigg(\mathbb{E}\Big[err(\boldsymbol{\theta}, \mathcal{B}_t, x)\Big] \mathbb{E}\Big[\sum_{x\in\mathcal{B}_t} \Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x) \cdot \mathbf{z}\Big] \\
&\qquad + cov\Big(err(\boldsymbol{\theta}, \mathcal{B}_t, \mathbf{z}), \sum_{x\in\mathcal{B}_t} \Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x) \cdot \mathbf{z}\Big)\Bigg) \\
=& |\mathcal{B}_t| \mathbb{E}\Big[err(\boldsymbol{\theta}, \mathcal{B}_t, \mathbf{z})\Big] \nabla \mathcal{L}(\boldsymbol{\theta}_t)^T \nabla \mathcal{L}(\boldsymbol{\theta}_t) \\
&+ \nabla \mathcal{L}(\boldsymbol{\theta}_t)^T cov\Big(err(\boldsymbol{\theta}, \mathcal{B}_t, \mathbf{z}), \sum_{x\in\mathcal{B}_t} \Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x) \cdot \mathbf{z}\Big) \\
\leq& \tau |\mathcal{B}_t| \cdot \|\mathcal{L}(\boldsymbol{\theta}_t)\|^2 \\
&+ \|\mathcal{L}(\boldsymbol{\theta}_t)\| \| cov\Big(err(\boldsymbol{\theta}, \mathcal{B}_t, \mathbf{z}), \sum_{x\in\mathcal{B}_t} \Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x) \cdot \mathbf{z}\Big) \|
\end{aligned}
$$

By Cauchy-Schwarz inequality, we have

$$
\begin{aligned}
&\| cov\Big(err(\boldsymbol{\theta}_t, \mathcal{B}_t, \mathbf{z}), \sum_{x\in\mathcal{B}_t} \Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x) \cdot \mathbf{z}\Big) \| \\
\leq& cov\Big(err(\boldsymbol{\theta}_t, \mathcal{B}_t, \mathbf{z})\Big) |\mathcal{B}_t| \sqrt{\lambda_{\max}(\Sigma)},
\end{aligned}
$$

where $\lambda_{\max}(\boldsymbol{\Sigma})$ is the maximum eigenvalue of matrix $\boldsymbol{\Sigma} = \text{cov}(\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x) \cdot \mathbf{z})$. We also have $\leq E[err(\boldsymbol{\theta}_t, \mathcal{B}_t, \mathbf{z})^2]$.

Plugging in equation 31, we have that

$$
\begin{aligned}
&\mathbb{E}[\mathcal{L}(\boldsymbol{\theta}_{t+1})|\boldsymbol{\theta}_t] - \mathcal{L}(\boldsymbol{\theta}_t) \\
\leq& -\frac{\eta|\mathcal{B}_t|}{b} \cdot (1 - \mathbb{E}[err(\boldsymbol{\theta}_t, \mathcal{B}_t, \mathbf{z})]) \cdot \|\nabla \mathcal{L}(\boldsymbol{\theta}_t)\|^2 \\
&+ \frac{\eta|\mathcal{B}_t|}{b} \cdot \sqrt{3} \mathbb{E}[(err(\boldsymbol{\theta}_t, \mathcal{B}_t, \mathbf{z})^2] \cdot \mathbb{E}[\|\nabla \mathcal{L}(\boldsymbol{\theta}_t; x)\|^2] \\
&+ \left(\frac{\eta|\mathcal{B}_t|}{b}\right)^2 \cdot \frac{(1+\sigma_m^2) \cdot l \cdot r \cdot c^2}{2}
\end{aligned}
$$

Next, we set the target batch size b to the actual batch size $|\mathcal{B}_t|$ for simplicity (otherwise there will be some multiplicative factors close to 1 which does not affect the overall landscape), as is done in prior work for analyzing subsampled DP algorithms [106]. Then we have $\frac{|\mathcal{B}_t|}{b} = 1$, and the proof is completed. □

Proof of equation 17.

$$
\begin{aligned}
\text{err}(\boldsymbol{\theta}_t, x, \mathbf{z}) \leq& \mathbb{E}_{x,\mathbf{z}}\left[\frac{|\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)|}{|\widehat{\Delta_{\mathbf{z}}}(\boldsymbol{\theta}_t; x)|}\right] - 1 = 1 \cdot \Pr[|\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)| \leq c] \\
&+ \frac{1}{c} \cdot \mathbb{E}[|\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)| \cdot \mathbf{1}\{|\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)| > c\}] - 1 \\
\leq& \frac{1}{c} \cdot \mathbb{E}[|\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)| \cdot \mathbf{1}\{|\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)| > c\}].
\end{aligned}
$$

□

Proof of equation 18. Applying Cauchy-Schwarz inequality ($E[|X||Y|] \leq E[|X|^2]E[|Y|^2]$) and Chebyshev's inequality, for any $c^2 > \mathbb{E}\big[|\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)|^2\big]$, we can bound

$$
\begin{aligned}
&\mathbb{E}[|\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)| \cdot \mathbf{1}\{|\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)| > c\}] && (33) \\
\leq& \sqrt{\mathbb{E}[|\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)|^2]} \sqrt{\Pr[|\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)| > c]} && (34) \\
\leq& \frac{\mathbb{E}\big[|\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)|^2\big]}{c}. && (35)
\end{aligned}
$$

□

Proof that $\mathbb{E}(|\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)|^2) = cov\big(\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)\big)$. For readability, we assume $\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)$ follows some distribution of mean μ_x and standard deviation σ_x, where the randomness is over the choice of $\mathbf{z}$, then it is easy to verify that $\mu_x = 0$ (see Proposition 3). By the law of total variance and plugging in $\mu_x = 0$, we have

$$
\begin{aligned}
\mathbb{E}\big[|\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)|^2\big] =& \Big(\mathbb{E}_{x,\mathbf{z}}[\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)]\Big)^2 && (36) \\
&+ \mathbb{E}_x[\text{var}_{\mathbf{z}}(\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)|x)] && (37) \\
&+ \text{var}_x\Big(\mathbb{E}_{\mathbf{z}}[\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)|x]\Big) && (38) \\
=& \mathbb{E}_x[\sigma_x^2]. && (39)
\end{aligned}
$$

□

Proof of equation 25. According to Lemma 3, we have

$$
\Pr\left[\|\boldsymbol{\Delta}_{\text{agg}}(\boldsymbol{\theta}_t; x)\| > \frac{\sigma_x}{\sqrt{K}} + \sqrt{\frac{3\sqrt{\log K}}{K\sqrt{K}}} \sqrt{U}\right] \leq 1 - \frac{1}{K^2}.
$$

Since the probability term is not larger than 1, the clipping error is smaller than 1 for any record x, we can write

$$err_{\text{agg}}(\boldsymbol{\theta}_t, x, \{\mathbf{z}_k\}_{k=1}^K) \leq \frac{\frac{\sigma_x}{\sqrt{K}} + \sqrt{\frac{3\sqrt{\log K}}{K\sqrt{K}}}\sqrt{U}}{\beta_{\text{clip}}/\sqrt{K}} + \frac{1}{K^2} \tag{40}$$

$$= \frac{\sigma_x}{\beta_{\text{clip}}} + \Theta\Big(\sqrt{\frac{\log K}{\sqrt{K}}}\Big). \tag{41}$$

□

Proof of Lemma 3. Recall Bernstein's inequality.

Lemma 9 (Bernstein's inequality)**.** *For a sequence of n independent random variables $Y_1, \ldots, Y_n$ with mean μ_j, variance $\max_j var(Y_j) \leq \sigma^2$, and suppose that $\max_j |Y_j - \mu_j| < Y_{max}$ almost surely for all j, then for any $t > 0$,*

$$\Pr\left[\sum_{j=1}^n Y_j > \sum_{j=1}^n \mu_j + t\right] \leq \exp\left(-\frac{t^2}{2n\sigma^2 + \frac{2}{3}tY_{max}}\right). \tag{42}$$

In particular, if we set $t = 3\sqrt{n\log n}\cdot Y_{\max}$, then we have

$$\Pr\big[\sum_{j=1}^n Y_j > \sum_{j=1}^n \mu_j + 3\sqrt{n\log n}\cdot Y_{\max}\big] \leq \exp\left(-\frac{9n\log n(Y_{\max})^2}{2n\sigma^2 + 2\sqrt{n\log n}(Y_{\max})^2}\right).$$

Since $Y_{\max} > \sigma$, and $n > \log n$ we have

$$\frac{9n\log n(Y_{\max})^2}{2n\sigma^2 + 2\sqrt{n\log n}(Y_{\max})^2} \geq \frac{9}{4}\log n.$$

Hence, if we set $t = 3\sqrt{n\log n}\cdot Y_{\max}$, then we have

$$\Pr\big[\frac{1}{n}\sum_{j=1}^n Y_j > \frac{1}{n}\sum_{j=1}^n \mu_j + \frac{3\sqrt{\log n}}{\sqrt{n}}\cdot Y_{\max}\big] \leq \frac{1}{n^2}.$$

Applying this to $\|\boldsymbol{\Delta}(x)\|_2^2$, we have

$$\Pr\left[K\|\boldsymbol{\Delta}(x)\|^2 > \mathbb{E}[(\Delta_{\mathbf{z}_j}(x))^2] + \frac{3\sqrt{\log K}}{\sqrt{K}}\cdot B\right] \leq \frac{1}{K^2},$$

where B satisfies $|(\Delta_{\mathbf{z}_j}(x))^2 - \mathbb{E}[(\Delta_{\mathbf{z}_j}(x))^2]| < B$ almost surely. Dividing both sides in the probability by n and taking the square root, we have

$$\Pr\left[\|\boldsymbol{\Delta}(x)\| > \sqrt{\frac{\mathbb{E}[(\Delta_{\mathbf{z}_j}(x))^2]}{K}} + \sqrt{\frac{3\sqrt{\log K}}{K\sqrt{K}}}\sqrt{B}\right] \leq \frac{1}{K^2},$$

since $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$ for $a, b \geq 0$. To complete this proof, we need to show that $\mathbb{E}[\Delta_{\mathbf{z}_j}(x)] = 0$ so that $\mathbb{E}[(\Delta_{\mathbf{z}_j}(x))^2] = var(\Delta(x))$, which follows from the following proposition.

Proposition 3. *$\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)$ follows a symmetric distribution. Namely, for any $w \in \mathbb{R}$, we have $\Pr[\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x) = w] = \Pr[\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x) = -w]$. We also have that the distribution has mean 0, and variance of $\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x)$ equals to $\mathbb{E}[(\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x))^2]$*

We sketch the proof of Propsition 3 as follows. With model parameter $\boldsymbol{\theta}_t$, record x, and perturbation scale ϕ fixed, we define function $f(\mathbf{v}) = \frac{\mathcal{L}(\boldsymbol{\theta}_t + \phi\mathbf{v}; x)}{2\phi}$. Then we can write $\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x) = f(\mathbf{z}) - f(-\mathbf{z})$. For any $u \in \mathbb{R}$, we have

$$\begin{aligned}
\Pr[\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x) = w] &= \int_v \Pr[f(\mathbf{z}) = v, f(-\mathbf{z}) = v - w]dv \\
&= \int_v \Pr[\mathbf{z} = \mathbf{y}; f(\mathbf{y}) = v, f(-\mathbf{y}) = v - w]dv \\
&= \int_v \Pr[\mathbf{z} = -\mathbf{y}; f(-\mathbf{y}) = v, f(\mathbf{y}) = v - w]dv \\
&= \int_v \Pr[\mathbf{z} = -\mathbf{y}; f(\mathbf{y}) = v - w, f(-\mathbf{y}) = v]dv.
\end{aligned}$$

Since $\mathbf{z}$ is sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I}_d)$, we have that $\Pr[\mathbf{z} = \mathbf{y}] = \Pr[\mathbf{z} = -\mathbf{y}]$ for any vector $\mathbf{y} \in \mathbb{R}^d$. Hence, we have

$$\begin{aligned}
\Pr[\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x) = w] &= \int_v \Pr[\mathbf{z} = \mathbf{y}; f(\mathbf{y}) = v - w, f(-\mathbf{y}) = v]dv \\
&= \Pr[\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x) = -w].
\end{aligned}$$

In addition, when $\phi \to 0$, then $\mathbb{E}[(\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x))^2] = \|\nabla\mathcal{L}(\boldsymbol{\theta}_t; x)\|^2$, since $\Delta_{\mathbf{z}}(\boldsymbol{\theta}_t; x) = \mathbf{z}^T\nabla\mathcal{L}(\boldsymbol{\theta}_t; x)$ when $\phi \to 0$. □

Proof of Lemma 4. Recall that $\sum_{k=1}^K \big(\sum_{x\in\mathcal{B}} \widehat{\Delta}_{\text{agg},k}(\boldsymbol{\theta}_t; x) + v_k\big)\cdot\mathbf{z}_k$ is used for DP-AggZO for updating the model, where $\widehat{\Delta}_{\text{agg},k}(\boldsymbol{\theta}_t; x)$ stands for the k-th dimension of the clipped aggregate vector and v_k is sampled from $\mathcal{N}(0, \sigma_m^2 c^2)$. Then we have

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}_{t+1}) \leq\ & \mathcal{L}(\boldsymbol{\theta}_t) + \nabla\mathcal{L}(\boldsymbol{\theta}_t)^T(\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t) \\
&+ \frac{1}{2}(\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t)^T\mathbf{H}(\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t)(\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t) \\
=\ & \mathcal{L}(\boldsymbol{\theta}_t) - \frac{\eta}{b}\cdot\nabla\mathcal{L}(\boldsymbol{\theta}_t)^T\sum_{k=1}^K\Big(\sum_{x\in\mathcal{B}}\widehat{\Delta}_{\text{agg},k}(\boldsymbol{\theta}_t; x)\Big)\cdot\mathbf{z}_k \\
&- \frac{\eta}{b}\cdot\nabla\mathcal{L}(\boldsymbol{\theta}_t)^T\sum_{k=1}^K v_k\cdot\mathbf{z}_k \\
&+ \sum_{k=1}^K\frac{\eta^2\Big(\big(\sum_{x\in\mathcal{B}}\widehat{\Delta}_{\text{agg},k}(\boldsymbol{\theta}_t; x)\big) + v_k\Big)^2}{2b^2}\cdot\mathbf{z}_k^T\mathbf{H}\mathbf{z}_k \\
&+ \sum_{k\neq k'}\frac{\eta^2}{2b^2}\cdot\Big(\big(\sum_{x\in\mathcal{B}}\widehat{\Delta}_{\text{agg},k}(\boldsymbol{\theta}_t; x)\big) + v_k\Big) \\
&\qquad\cdot\Big(\big(\sum_{x\in\mathcal{B}}\widehat{\Delta}_{\text{agg},k'}(\boldsymbol{\theta}_t; x)\big) + v_{k'}\Big)\cdot\mathbf{z}_k^T\mathbf{H}\mathbf{z}_{k'}
\end{aligned}$$

We define $err_{\text{agg}}(\boldsymbol{\theta}_t, x, \{\mathbf{z}_k\}_{k=1}^K) = \frac{\|\Delta_{\text{agg}}(\boldsymbol{\theta}_t; x) - \widehat{\Delta}_{\text{agg}}(\boldsymbol{\theta}_t; x)\|}{\|\Delta_{\text{agg}}(\boldsymbol{\theta}_t; x)\|}$. For readability, we abbreviate it as err_{agg}. It is easy to verify that err_{agg} is in the range of $[0, 1)$ for clipping thresholds larger than 0.

We then take the conditional expectation on both sides with respect to $\boldsymbol{\theta}_t$ and get

$$
\begin{aligned}
&\mathbb{E}[\mathcal{L}(\boldsymbol{\theta}_{t+1})|\boldsymbol{\theta}_t)] - \mathcal{L}(\boldsymbol{\theta}_t) \\
&\leq -\frac{\eta|\mathcal{B}_t|}{b}\cdot\nabla\mathcal{L}(\boldsymbol{\theta}_t)^T\mathbb{E}\Big[(1-err_{\text{agg}})\sum_{k=1}^{K}\Delta_{\text{agg},k}(\boldsymbol{\theta}_t;x)\cdot\mathbf{z}_k\Big] \\
&+\mathbb{E}\Big[\sum_{k=1}^{K}\frac{\eta^2\big(\sum_{x\in\mathcal{B}}\widehat{\Delta}_{\text{agg},k}(\boldsymbol{\theta}_t;x)\big)^2}{2b^2}\Big]\cdot lr + \frac{\eta^2}{2b^2}\cdot K\sigma_m^2c^2\cdot lr \\
&\leq -\frac{\eta|\mathcal{B}_t|}{b}\cdot\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|^2 \\
&+\frac{\eta|\mathcal{B}_t|}{b}\cdot\nabla\mathcal{L}(\boldsymbol{\theta}_t)^T\mathbb{E}\Big[err_{\text{agg}}\sum_{k=1}^{K}\Delta_{\text{agg},k}(\boldsymbol{\theta}_t;x)\cdot\mathbf{z}_k\Big] \\
&+\frac{\eta^2|\mathcal{B}_t|}{2b^2}\cdot\mathbb{E}\Big[\sum_{k=1}^{K}|\mathcal{B}_t|\sum_{x\in\mathcal{B}_t}\big(\widehat{\Delta}_{\text{agg},k}(\boldsymbol{\theta}_t;x)\big)^2\Big] \\
&+\frac{\eta^2}{2b^2}\cdot K\sigma_m^2c^2\cdot lr,
\end{aligned}
$$

where c is the clipping threshold applied to the norm of the aggregate vector $\boldsymbol{\Delta}_{\text{agg}}(\boldsymbol{\theta}_t;x)$ and we have $\mathbb{E}\Big[\sum_{k=1}^{K}|\mathcal{B}_t|\sum_{x\in\mathcal{B}_t}\big(\widehat{\Delta}_{\text{agg},k}(\boldsymbol{\theta}_t;x)\big)^2\Big] \leq |\mathcal{B}_t|c^2$. For $\mathbb{E}\Big[err_{\text{agg}}\sum_{k=1}^{K}\Delta_{\text{agg},k}(\boldsymbol{\theta}_t;x)\cdot\mathbf{z}_k\Big]$, we apply the same trick as in the proof for Lemma 2, getting

$$
\|cov\Big(err_{\text{agg}},\sum_{k=1}^{K}\Delta_{\text{agg},k}(\boldsymbol{\theta}_t;x)\cdot\mathbf{z}_k\Big)\| \leq cov(err_{\text{agg}})\sqrt{\frac{1}{K}\lambda_{\max}(\Sigma)},
$$

where the extra factor of $\frac{1}{\sqrt{K}}$ comes from the fact that the latter vector is the average of K independent samples. According to equation 25, we have that for clipping threshold $c = \beta_{\text{clip}}\cdot\frac{1}{\sqrt{K}}$ for some β_{clip}, with probability at least $1-\frac{1}{K^2}$, the clipping error is bounded as

$$
err_{\text{agg}}(\boldsymbol{\theta}_t,x,\{\mathbf{z}_k\}_{k=1}^K) \leq \frac{\sigma_x}{\beta_{\text{clip}}} + \Theta\left(\left(\frac{\log K}{K}\right)^{\frac{1}{4}}\right). \tag{43}
$$

We denote A_t as the event that $err_{\text{agg}}(\boldsymbol{\theta}_t,x,\{\mathbf{z}_k\}_{k=1}^K) \leq \frac{\sigma_x}{\beta_{\text{clip}}} + \Theta\left(\left(\frac{\log K}{K}\right)^{\frac{1}{4}}\right)$ holds, and let $\bar{A}_t$ be the event when A does not happen. $P(A_t)+P(\bar{A}_t)=1$. Then we can rewrite

$$
\mathbb{E}[err_{\text{agg}}(\boldsymbol{\theta}_t,x,\{\mathbf{z}_k\}_{k=1}^K)] \leq \frac{\sigma_x}{\beta_{\text{clip}}} + \Theta\left(\left(\frac{\log K}{K}\right)^{\frac{1}{4}}\right) + \Theta(\frac{1}{K^2}). \tag{44}
$$

As K increases, the Big-Theta terms converge to 0. □

USENIX Security '25 Artifact Appendix: Unlocking the Power of Differentially Private Zeroth-order Optimization for Fine-tuning LLMs

Ergute Bao[*], *Yangfan Jiang*[†], *Fei Wei*[*], *Xiaokui Xiao*[†], *Zitao Li*[*], *Yaliang Li*[*], *Bolin Ding*[*]
[*]*Alibaba Group,* [†]*National University of Singapore*

A Artifact Appendix

A.1 Abstract

The main idea of our proposed method, DP-AggZO, is to aggregate **multiple zeroth-order estimates** for the exact gradients, computed over independent perturbation vectors (random Gaussian vectors), before enforcing differential privacy (i.e., artificial clipping, taking the average, and then injecting random DP noises). Compared with the vanilla DPZO (or DPZero), which is effectively a degenerated version of DP-AggZO with only **one** zeroth-order estimate, our DP-AggZO achieves much better utility under the same privacy constraints. Our DP-AggZO also outperforms the state-of-the-art DP-AdamW in some cases. This artifact is used for validating the above claim.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

Security, privacy, and ethical concerns are not applicable to this artifact as the models and datasets are publicly available.

A.2.2 How to access

Access via Zenodo in https://zenodo.org/records/15594622.

A.2.3 Hardware dependencies

A workstation or cloud computing node with a GPU (preferrably with GPU memory larger than 20 GB), e.g., RTX 4090 GPU 24GB, or above (larger GPU memory is needed if run larger models, e.g., OPT 6.7B).

A.2.4 Software dependencies

Linux system Ubuntu 22.04.4, installed with python 3.9.18, with torch==2.4.0+cu121, transformers==4.28.1, and opacus==1.4.0. More on enviroments can be found in the file named "environments.yml" provided.

A.2.5 Benchmarks

We evaluate on RoBERTa (355M) [1], which is a pretrained model on English language using a masked language modeling (MLM) objective, and OPT-1.3B and OPT-6.7B [2], which are parts of the a suite of decoder-only pre-trained transformers ranging from 125M to 175B parameters.

In the provided script, there are on six datasets for different classification tasks, including SST-2 and SST-5 [3] for sentiment analysis (i.e., determine if the given text is positive/negative), SNLI [4], MNLI [5], and RTE [6] for natural language inference (i.e., determine if the given premise and hypothesis are in the relationship of entailment/neutral/contradiction), and TREC [7] for topic assignment (i.e., determine which topic the given question falls under). For each dataset, we generate 512 samples for each class for training.

For larger models OPT-1.3B and OPT-6.7B that require more resources to fine-tune, we focus on one classification task SST-2 and one generation task SQuAD [8] (the fine-tuned model answers to a given question containing relevant contexts). For each dataset, we use 1000 samples for training.

A.3 Set-up

A.3.1 Installation

Please Use the file named "environments.yml" provided to install the environment using pip or conda. For testing RoBERTa (355M), go to folder roberta and install the dataset as specified in the readme file provided.

A.3.2 Basic Test

After installation of the environment and datasets, to test the functionality, run the following script in bash.

[1] https://huggingface.co/FacebookAI/roberta-large
[2] https://huggingface.co/facebook/opt-1.3b
[3] https://aclanthology.org/D13-1170/
[4] https://aclanthology.org/D15-1075/
[5] https://aclanthology.org/N18-1101/
[6] https://dl.acm.org/doi/10.1007/11736790_9
[7] https://aclanthology.org/L00-1018/
[8] https://aclanthology.org/D16-1264/

```
CUDA_VISIBLE_DEVICES=0 DPZERO_PRIVACY_EPS=2
    DP_SAMPLE_RATE=0.0208 STEP=500 \
SEED=42 NUM_DIRECTION=64
    RANDOM_DIRECTION_SEED=100 LR=6e-4 \
DPZERO_THRESHOLD=1 TASK="MNLI" bash examples
    /dpaggzo.sh
```

This script takes around 80 minutes to run on a RTX 4090 GPU and gives a test accuracy around 72%, which is much better than the utility of the original DPZO/DPZero under the same privacy constraint (around 65%, or refer to Table 2 on Page 9 of their original paper [9]).

Results on other datasets can be obtained by changing the "TASK" parameter. We also provided more examples in the "readme" file.

A.4 Evaluation workflow

A.4.1 Major Claims

We make two major claims.

(C1): *Our DP-AggZO can outperform the vanilla DPZero/DPZO in terms of test accuracy under the same privacy constraints.*

(C2): *Our DP-AggZO can sometimes outperform DP-AdamW in terms of test accuracy under the same privacy constraints.*

A.4.2 Experiments

C1-1: Reproducing DP-AggZO results for MNLI with privacy level $\varepsilon = 2$. You can run the DP-AggZO experiments for the MNLI task directly using the following commands:

```
CUDA_VISIBLE_DEVICES=0 DPZERO_PRIVACY_EPS=2
    DP_SAMPLE_RATE=0.0416 STEP=1000 \
SEED=42 NUM_DIRECTION=64
    RANDOM_DIRECTION_SEED=100 LR=8e-5 \
DPZERO_THRESHOLD=5 TASK="MNLI" bash examples
    /dpaggzo.sh
```

or

```
CUDA_VISIBLE_DEVICES=0 DPZERO_PRIVACY_EPS=2
    DP_SAMPLE_RATE=0.0416 STEP=1000 \
SEED=42 NUM_DIRECTION=64
    RANDOM_DIRECTION_SEED=100 LR=5e-4 \
DPZERO_THRESHOLD=1 TASK="MNLI" bash examples
    /dpaggzo.sh
```

Expected outcome:

- Compute time: 5 hours on RTX 4090, 14 hours on RTX A5000.
- Test accuracy is ∼74% on RTX A5000 or H20 GPU; ∼71% on RTX 4090 GPU.
- Significantly better than vanilla DPZO/DPZero under the same privacy level (see below).

C1-2: Reproducing DP-AggZO results with $K = 1$ (equivalent to DPZero/DPZO baseline) with privacy level $\varepsilon = 2$.

```
CUDA_VISIBLE_DEVICES=0 DPZERO_PRIVACY_EPS=2
    DP_SAMPLE_RATE=0.0416 STEP=5000 \
SEED=42 NUM_DIRECTION=1
    RANDOM_DIRECTION_SEED=100 LR=2e-6 \
DPZERO_THRESHOLD=200 TASK="MNLI" bash
    examples/dpaggzo.sh
```

Expected outcome:

- Compute time: 40 minutes on RTX 4090
- Test accuracy: 65% on RTX 4090 GPU, or can refer to the original paper: arxiv.org/pdf/2310.09639

Combining the results of **C1-1** and **C1-2**, we can verify that DP-AggZO outperforms the vanilla DPZO/DPZero under the same privacy constraints.

C2-1: Reproducing DP-AggZO results for MNLI with privacy level $\varepsilon = 0.5$. You can run the DP-AggZO experiments for the MNLI task directly using the following command:

```
CUDA_VISIBLE_DEVICES=0 DPZERO_PRIVACY_EPS
    =0.5 DP_SAMPLE_RATE=0.0416 STEP=500 \
SEED=42 NUM_DIRECTION=64
    RANDOM_DIRECTION_SEED=100 LR=2e-4 \
DPZERO_THRESHOLD=1 TASK="MNLI" bash examples
    /dpaggzo.sh
```

Expected outcome:

- Compute time: 3 hours on RTX 4090, 7 hours on RTX A5000.
- Accuracy: ∼63.5% on H20 and A5000 GPUs
- Better than DP-AdamW under the same privacy level (∼ 62%) (see below).

C2-2: Reproducing the result on DP-AdamW using the following fommand:

```
CUDA_VISIBLE_DEVICES=0 DP_SAMPLE_RATE=0.0416
     STEP=1000 SEED=42 LR=1e-4 \
DPSGD_THRESHOLD=10 DPSGD_PRIVACY_EPS=0.5
    DPSGD_PRIVACY_DELTA=1e-5 \
TASK="MNLI" bash examples/dpsgd.sh
```

The results from **C2-1** and **C2-2** demonstrate that **DP-AggZO can outperform DP-AdamW** the same privacy constraints, but the improvement is not as significant as that for DPZO/DPZero. We refer to the "readme" file provided for more examples.

To use this artifact beyond the models presented in

[9] https://arxiv.org/pdf/2310.09639

A.5 Notes on Reusability

To use this artifact beyond the models presented in this paper, we would recommend refactoring the code based on the latest implementation and algorithms of the model of interest. Some functions/libraries may become obsolete in the future while the general algorithmic idea of using multiple independent zeroth-order estimates to reduce clipping error could still apply.

A.6 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2025/.