TOWARDS HYPERPARAMETER-FREE OPTIMIZATION WITH DIFFERENTIAL PRIVACY

Anonymous authors
Paper under double-blind review

ABSTRACT

Differential privacy (DP) is a privacy-preserving paradigm that protects the training data when training deep learning models. Critically, the performance of models is determined by the training hyperparameters, especially those of the learning rate schedule, thus requiring fine-grained hyperparameter tuning on the data. In practice, it is common to tune the learning rate hyperparameters through the grid search that (1) is computationally expensive as multiple runs are needed, and (2) increases the risk of data leakage as the selection of hyperparameters is data-dependent. In this work, we adapt the automatic learning rate schedule to DP optimization for any models and optimizers, so as to significantly mitigate or even eliminate the cost of hyperparameter tuning when applied together with automatic per-sample gradient clipping. Our hyperparamter-free DP optimization is almost as computationally efficient as the standard non-DP optimization, and achieves state-of-the-art DP performance on various language and vision tasks.

1 INTRODUCTION

The performance of deep learning models relies on a proper configuration of training hyperparameters. In particular, the learning rate schedule is critical to the optimization, as a large learning rate may lead to divergence, while a small learning rate may slowdown the converge too much to be useful. In practice, people have used heuristic learning rate schedules that are controlled by many hyperparameters. For example, many large language models including LLAMA2 Touvron et al. (2023) uses linear warmup and cosine decay in its learning rate schedule, which are controlled by 3 hyperparameters. Generally speaking, hyperparameter tuning (especially for multiple hyperparameters) can be expensive for large datasets and large models.

To address this challenge, it is desirable or even necessary to determine the learning rate schedule in an adaptive and data-dependent way, without little if any manual effort. Recent advances such as D-adaptation Defazio & Mishchenko (2023), Prodigy Mishchenko & Defazio (2023), DoG Ivgi et al. (2023), DoWG Khaled et al. (2023), U-DoG Kreisler et al. (2024), and GeN Bu et al. (2023b) have demonstrated the feasibility of automatic learning rate schedule, with some promising empirical results in deep learning (see a detailed discussion in Section 2.3).

While these learning rate methods are evolving in the standard non-DP regime, their adaptive dependency on the data can raise the privacy risks of memorizing and reproducing the training data. As an example, we consider the ZO-SGD optimizer

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_{\text{ZO-SGD}} \boldsymbol{z}_t$$

where $\boldsymbol{w}_t$ is the model parameters, $\boldsymbol{z}_t$ is a random vector that is independent of any data, and $\eta_{\text{ZO-SGD}} \approx \frac{\partial L}{\partial \boldsymbol{w}_t}^\top \boldsymbol{z}_t \in \mathbb{R}$ is the effective learning rate, with L being the loss value. In Malladi et al. (2023), ZO-SGD can effectively optimize models such as OPT 13-60B in the few-shot and many-shot settings. Hence, by using an adaptive hyperparameter $\eta_{\text{ZO-SGD}}$, the models are able to learn and possibly memorize the training data even if the descent direction is data-independent. In fact, the private information can also leak through other hyperparameters and in other models, where an outlier datapoint can be revealed via the membership inference attacks in support vector machinesPapernot & Steinke (2021).

Specifically, in the DP deep learning regime, the privacy risks from the non-DP hyperparameter tuning could render the privacy guarantee non-rigorous. In practice, most work has leveraged the DP

optimizers Abadi et al. (2016) to offer the privacy guarantee, where the privatization is only on the gradients, but not on the hyperparameters such as the clipping threshold R_g and the learning rate η. A common approach is to trial-and-error on multiple (R_g, η) pairs and select the best hyperparameters, as showcased by Li et al. (2021); Kurakin et al. (2022).

At high level, there are two approaches to accommodate the privacy risk in hyperparameter tuning: (1) The more explored approach is to assign a small amount of privacy budget to privatize the hyperparameter tuning. Examples include DP-Hypo Liu & Talwar (2019); Papernot & Steinke (2021); Wang et al. (2023) and DP-ZO-SGD Tang et al. (2024); Liu et al. (2024); Zhang et al.. However, these methods may suffer from worse performance due to larger DP noise addition from the reduced privacy budget, or high computation overhead. (2) The less explored approach is to adopt hyperparameter-free methods. For instance, Bu et al. (2023b); Yang et al. (2022) replace the per-sample gradient clipping with the per-sample gradient normalization (i.e. setting $R_g = 0$), so as to remove the need to tune the hyperparameter R_g in DP optimizers.

In this work, we work towards the hyperparameter-free optimization with DP, by adapting learning-rate-free methods in DP optimization:

$$\text{DP-SGD} \Longrightarrow \begin{cases} \text{Vanilla:} & \boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_{\text{manual}} \left(\sum_{i \in \mathcal{B}} \min\{ \frac{R_g}{||\boldsymbol{g}_i||}, 1 \} \boldsymbol{g}_i + \sigma R_g \boldsymbol{z} \right) \\ \text{Hyperparameter-free:} & \boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_{\text{GeN-DP}} \left(\sum_{i \in \mathcal{B}} \frac{\boldsymbol{g}_i}{||\boldsymbol{g}_i||} + \sigma \boldsymbol{z} \right) \end{cases} \tag{1}$$

Our contributions are summarized as follows: 1) We propose **HyFreeDP**, a hyperparameter-free DP framework that rigorously guarantees DP on the hyperparameter tuning of (R_g, η_t), and works with any optimizer. 2) We apply the loss privatization to leverage GeN learning rate under DP, with a specific auto-regressive clipping threshold R_l that aims to minimize the clipping bias. 3) We give an end-to-end privacy accounting method that adds $< 1\%$ more noise on the gradient, while accurately capturing the loss curvature to determine the adaptive learning rate. 4) We show the strong performance and high efficiency of our method empirically.

2 Preliminaries and Related works

2.1 Differentially Private Optimization

Overview of DP optimization. We aim to minimize the loss $\sum_i L(\boldsymbol{w}, \boldsymbol{x}_i)$ where $\boldsymbol{w} \in \mathbb{R}^d$ is the model parameters and $\boldsymbol{x}_i$ is one of data points with $1 \leq i \leq N$. We denote the per-sample gradient as $\boldsymbol{g}_i(\boldsymbol{w}) := \frac{\partial L(\boldsymbol{w}, \boldsymbol{x}_i)}{\partial \boldsymbol{w}} \in \mathbb{R}^d$ and the mini-batch gradient at the t^{th} updating iteration as $\boldsymbol{m}_t \in \mathbb{R}^d$: for a batch size $B \leq N$,

$$\boldsymbol{m}_t(\{\boldsymbol{g}_i\}_{i=1}^B; R_g, \sigma_g) := [\sum_i^B c_i(R_g) \boldsymbol{g}_i + \sigma_g R_g \cdot \boldsymbol{z}_g]/B \tag{2}$$

where $\boldsymbol{z}_g \sim \mathcal{N}(0, \mathbf{I}_d)$ is the Gaussian noise, and $c_i(R_g) = \min(R_g/|\boldsymbol{g}_i|, 1)$ is the per-sample gradient clipping factor in Abadi et al. (2016); De et al. (2022) with R_g being the clipping threshold.

In particular, $\boldsymbol{m}_t$ reduces to the standard non-DP gradient $\sum_i \boldsymbol{g}_i/B$ when $\sigma_g = 0$ and R_g is sufficiently large (i.e., no clipping is applied). The clipped and perturbed batch gradient becomes the DP gradient $\boldsymbol{m} \equiv \boldsymbol{m}_{\text{DP}}$ whenever $\sigma_g > 0$, with stronger privacy guarantee for larger σ_g. On top of $\boldsymbol{m}$, models can be optimized using any optimizer such as SGD and Adam through

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t \cdot \mathbf{G}_t(\boldsymbol{m}\{\boldsymbol{g}_i\}) \tag{3}$$

in which $\mathbf{G}_t$ is the post-processing such as momentum, adaptive pre-conditioning, and weight decay.

Hyper-parameter matters for DP optimization. Previous works De et al. (2022); Li et al. (2021) reveal that the performance of DP optimization is sensitive to the hyper-parameter choices. On the one hand, DP by itself brings extra hyper-parameters, such as the gradient clipping threshold R_g, making the tuning more complex. An adaptive clipping method Andrew et al. (2021) proposes to automatically learn R_g at each iteration, with an extra privacy budget that translates to worse accuracy. While there are methods that does not incur additional privacy budget. Automatic clipping

(or per-sample normalization, or AutoClip) Bu et al. (2023b); Yang et al. (2022) is a technique that uses $c_i = 1/||\boldsymbol{g}_i||$ to replace the R_g-dependent per-sample clipping, and thus removes the hyperparameter R_g from DP algorithms. On the other hand, hyper-parameter tuning for DP training has different patterns than Non-DP training and cannot borrow previous experience on Non-DP. For example, previous works Li et al. (2021); De et al. (2022) observe that there is no benefit from decaying the learning rate during training and the optimal learning rate is much higher than the optimal one in Non-DP training.

2.2 END-TO-END DP GUARANTEE FOR OPTIMIZATION AND TUNING

However, hyper-parameter tuning enlarges privacy risks Papernot & Steinke (2021), and it is necessary to provide end-to-end privacy guarantee for DP. There are two existing technical paths to solve this problem: 1) Guaranteeing DP for the entire tuning and training process Mohapatra et al. (2022); Papernot & Steinke (2021); Wang et al. (2023); Liu & Talwar (2019), which either requires prior knowledge or consumes significant privacy budget. For instance, Liu & Talwar (2019) showed that repeated searching with random iterations satisfies $(3\epsilon, 0)$-DP if each run was $(\epsilon, 0)$-DP. 2) Making DP optimization hyper-parameter free through automatic clipping Bu et al. (2023b); Yang et al. (2022), though learning rate tuning η remains necessary as clipping coefficients are absorbed into it. Our work follows the second approach, aiming to resolve tuning on η—the last critical hyper-parameter—by finding a universal configuration across datasets and tasks.

Following the framework of previous works Liu & Talwar (2019); Papernot & Steinke (2021), suppose there are m privacy-preserving training algorithms $\mathcal{M}_1, \cdots, \mathcal{M}_m$ which corresponds to m possible hyper-parameters. Denoting the whole process of training and hyper-parameter tuning as $\mathcal{M}$, it takes input as a training dataset D and outputs the best outcome over a finite set of m possible hyper-parameters by running $\mathcal{M}_1, \cdots, \mathcal{M}_m$. The end-to-end DP ensures that $\mathcal{M}$ satisfies (ϵ, δ)-DP with respect to any neighboring datasets and any outcome. And the outcome includes the best model parameters and the corresponding hyper-parameters. In our hyper-parameter-free solution, there is a single training with $m = 1$, and the output hyper-parameters are data-independent.

2.3 AUTOMATIC LEARNING RATE SCHEDULE

Automatic learning rate schedules (also known as learning-rate-free or parameter-free methods) have demonstrated promising performance in deep learning, with little to none manual efforts to select the learning rate. Specifically, D-adaptation Defazio & Mishchenko (2023), Prodigy Mishchenko & Defazio (2023), and DoG Ivgi et al. (2023) (and its variants) have proposed to estimate $\eta \approx \frac{D}{G\sqrt{T}}$ where $D = ||\boldsymbol{w}_0 - \boldsymbol{w}_*||$ is the initialization-to-minimizer distance, G is the Lipschitz continuity constant, and T is the total number of iterations. These methods have their roots in the convergence theory under the convex and Lipschitz conditions, and may not be accurate when applied in the DP regime when the gradient is noisy. In fact, the estimation of D and G can deviate significantly from the truth when ϵ is small, i.e. the DP noise in gradient is large, as shown in Table 2.

Along an orthogonal direction, GeN Bu & Xu (2024) leverages the Taylor approximation of Hessian information to set the learning rate η_{GeN}, without assuming the Lipschitz continuity or the knowledge of D. Given any descent vector $\boldsymbol{m}$[1], we re-write their quadratic function with our notations in (4):

$$\eta_{\text{GeN}}(\boldsymbol{m}) := \frac{\widetilde{\mathbf{G}}^\top \boldsymbol{m}}{\boldsymbol{m}\widetilde{\mathbf{H}}\boldsymbol{m}} = \text{argmin}_\eta L(\boldsymbol{w}) - \widetilde{\mathbf{G}}^\top \boldsymbol{m}\eta + \boldsymbol{m}\widetilde{\mathbf{H}}\boldsymbol{m}\frac{\eta^2}{2} \approx \text{argmin}_\eta L(\boldsymbol{w} - \eta\boldsymbol{m}) \quad (4)$$

Numerically, η_{GeN} can be computed up to any precision by curve fitting or finite difference. Under the non-DP regime, given a series of η_i and loss values $L(\boldsymbol{w} - \eta_i\boldsymbol{m})$,Bu & Xu (2024) obtains the numerator and denominator of η_{GeN} by solving the problem in Equation (4):

$$\boldsymbol{m}^\top\widetilde{\mathbf{H}}\boldsymbol{m}, \widetilde{\mathbf{G}}^\top\boldsymbol{m} \approx \text{argmin}_{a,b} \sum_i \left| L(\boldsymbol{w} - \eta_i\boldsymbol{m}) - \left(L(\boldsymbol{w}) - b\eta_i + a\frac{\eta_i^2}{2} \right) \right|^2 \quad (5)$$

Nevertheless, directly applying these non-DP automatic learning rate scheduler with DP gradient in (2) and using $\eta_{\text{GeN}}(\boldsymbol{m})$ will violate DP because the learning rate estimation is obtained from forward passes on batches of private data. We defer the explanation and solution to Section 3.

[1]We omit the iteration index t for a brief notation.

3 LOSS VALUE PRIVATIZATION WITH MINIMAL CLIPPING BIAS

3.1 PRIVATIZED QUADRATIC FUNCTION

We emphasize that the learning rate $\eta_{\text{GeN}}(\boldsymbol{m})$ is not DP, because even though $\boldsymbol{m}$ is privatized, the data is accessed without protection through $\widetilde{\mathbf{G}}$ and $\widetilde{\mathbf{H}}$ in (5). To solve this issue, we introduce a privatized variant of (5),

$$(\boldsymbol{m}^\top \widetilde{\mathbf{H}} \boldsymbol{m})_{\text{DP}}, (\widetilde{\mathbf{G}}^\top \boldsymbol{m})_{\text{DP}} := \text{argmin}_{a,b} \sum_i \left| \tilde{L}(\boldsymbol{w} - \eta_i \boldsymbol{m}_{\text{DP}}) - \left(\tilde{L}(\boldsymbol{w}) - b\eta_i + a\frac{\eta_i^2}{2} \right) \right|^2 \tag{6}$$

which not only replaces $\boldsymbol{m}$ with the DP gradient $\boldsymbol{m}_{\text{DP}}$ but also privatizes the loss by $\tilde{L}(\boldsymbol{w} - \eta_i \boldsymbol{m}_{\text{DP}})$ as in (7). The resulting learning rate is $\eta_{\text{GeN-DP}} = \frac{(\widetilde{\mathbf{G}}^\top \boldsymbol{m})_{\text{DP}}}{(\boldsymbol{m}^\top \widetilde{\mathbf{H}} \boldsymbol{m})_{\text{DP}}}$, which is DP since every quantity in (6) is DP and because of the post-processing property. We now discuss the specifics of the loss value privatization $\tilde{L}(\boldsymbol{w} - \eta_i \boldsymbol{m}) \in \mathbb{R}$. In Table 1, we emphasize that the loss privatization is distinctively different from the gradient privatization because the loss is scalar, whereas the gradient is high-dimensional.

Table 1: Difference between the privatization of loss and gradient.

Aspects	Loss Privatization	Gradient Privatization
Dimension	1	d
Clipping Norm	L2 or L1	L2
Noise Magnitude	$\sqrt{\frac{2}{\pi}}\frac{\sigma_l R_l}{B}$	$\sqrt{d}\frac{\sigma_g R_g}{B}$
Key to Convergence	Clipping Bias	Noise Magnitude
Per-sample Operation	Clipping ($R_l \approx L$)	Normalization ($R_g \approx 0^+$)

From the perspective of per-sample clipping, the gradient is ubiquitously clipped on L2 norm, because $||\boldsymbol{g}_i||_2 \ll ||\boldsymbol{g}_i||_1$ in large neural networks, and consequently a Gaussian noise is added to privatize the gradient. In contrast, we can apply L2 or L1 norm for the loss clipping, and add Gaussian (by default) or Laplacian noise to the loss, respectively.

From the perspective of noising, the expected noise magnitude for loss privatization is $\mathbb{E}|z_l|\frac{\sigma_l R_l}{B} = \sqrt{\frac{2}{\pi}}\frac{\sigma_l R_l}{B}$ where $z_l \sim N(0,1)$ is the noise on loss, and that for gradient privatization is $\mathbb{E}||\boldsymbol{z}_g||\frac{\sigma_g R_g}{B} \approx \sqrt{d}\frac{\sigma_g R_g}{B}$ where the gradient noise vector $\boldsymbol{z}_g \sim N(0, \mathbf{I}_d)$ by the law of large numbers. On the one hand, the gradient noise $\boldsymbol{z}_g$ is large and requires small R_g to suppress the noise magnitude, as many works have use very small R Li et al. (2021); De et al. (2022). This leads to the automatic clipping in Bu et al. (2023b) when the gradient clipping effectively becomes the gradient normalization as $R_g \to 0^+$. In fact, each per-sample gradient (as a vector) has a magnitude and a direction, and the normalization neglects some if not all magnitude information about the per-sample gradients. On the other hand, we must not use a small R_l for the loss clipping because the per-sample loss (as a scalar) only has the magnitude information. We will show by Theorem 1 that the choice of threshold R_l creates a bias-variance tradeoff between the clipping and the noising for the loss privatization:

$$\tilde{L} = \frac{1}{B}\left[\sum_i \min(\frac{R_l}{L_i}, 1) L_i + \sigma_l R_l \cdot N(0,1) \right] \tag{7}$$

We note (7) is a private mean estimation that has been studied in previous works Biswas et al. (2020); Kamath et al. (2020), though many use an asymptotic threshold like $R_l + O(\log B)$, whereas Theorem 1 is more suited for our application in practice.

3.2 BIAS-VARIANCE TRADE-OFF IN LOSS PRIVATIZATION

Theorem 1. *The per-sample clipping bias of* (7) *is*

$$\left| \mathbb{E}(\tilde{L}) - \frac{\sum_i L_i}{B} \right| = \left| \frac{1}{B} \sum_i \min(\frac{R_l}{L_i}, 1) L_i - \frac{1}{B} \sum_i L_i \right| = \left| \frac{1}{B} \sum_i (L_i - R_l) \mathbb{I}(L_i > R_l) \right|$$

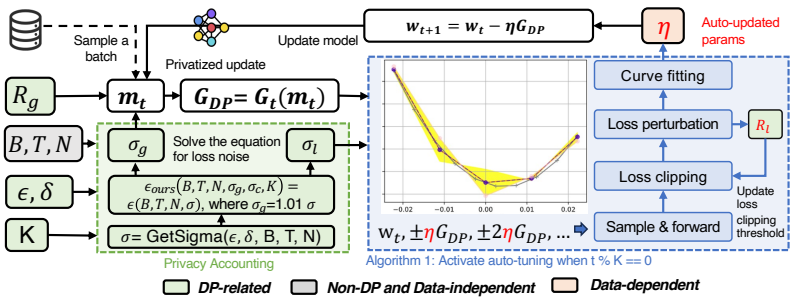

Figure 1: HyFreeDP overview with three types of hyper-parameters in the DP training. HyFreeDP saves tuning efforts via automatically tuning hyper-parameters in red text, and sets other parameters as default constants. We showcase with 5 points in curve fitting.

which is monotonically decreasing in R_l, and converges to $[\mathbb{E}(L_i|L_i > R_l) - R_l] \cdot \mathbb{P}(L_i > R_l)$ as $B \to \infty$. In contrast, the noise variance is $\mathrm{Var}(\tilde{L}) = (\sigma_l R_l / B)^2$ which is increasing in R_l.

In words, a large R_l reduces the clipping bias but magnifies the noise, and vice versa for a small R_l. We propose to use $R_l \approx L$ so that the clipping bias is close to zero (i.e. $\tilde{L}$ is approximately unbiased), and the loss noise shown in Table 1 is reasonably small for large batch size.

To put this into perspective, we give the explicit form of clipping bias when L_i follows a Gaussian distribution in Corollary 1.

Corollary 1. *Suppose $L_i \sim N(\mu, \xi^2)$, then the asymptotic clipping bias in Theorem 1 is*

$$[\mathbb{E}(L_i|L_i > R_l) - R_l] \cdot \mathbb{P}(L_i > R_l) = \xi[\phi(\alpha) - \alpha(1 - \Phi(\alpha)], \tag{8}$$

where $\alpha = \frac{R_l - \mu}{\xi}$, ϕ is the probability density function and Φ is the cumulative distribution function of standard normal distribution. The term (8) is strictly decreasing in α as well as in R_l.

4 Algorithm

4.1 Hyperparameter-free DP optimization

We present our algorithm in Algorithm 1, which is DP as guaranteed in Theorem 2, almost as efficient as the standard non-DP optimization by Section 4.4, and highly accurate and fast in convergence as demonstrated in Section 5. Importantly, we have split the hyperparameters into three classes, as shown in Figure 1: (1) **DP-related** hyperparameters that do not depend on the tasks, such as the gradient noise σ_g, the loss noise σ_l, and the update interval K, can be set as default constants For example, we fix $R_g \to 0^+$ and re-scale the learning rate by $1/R_g$ according to Auto Clipping and set $K = 5$. (2) Training hyperparameters that are robust to different models and datasets, which we view as **data-independent**, need-not-to-search, and not violating DP, such as the batch size B, the number of iterations T. We can set with default values based on Non-DP training experience or it is given as the dataset size N. We also fix other hyperparameters not explicitly displayed in Algorithm 1, e.g. throughout this paper, we fix the momentum coefficients and weight decay in AdamW at $(\beta_1, \beta_2, \text{weight\_decay}) = (0.9, 0.999, 0.01)$, which is the default in Pytorch. (3) Training hyperparameters that are **data-dependent**, which requires dynamical searching under DP, such as R_l and η. For these hyperparameters, Algorithm 1 adopts multiple *auto-regressive* designs, i.e. the variables to use in the t-th iteration is based on the $(t-1)$-th iteration, which has already been privatized. These auto-regressive designs allow new variables to preserve DP by the post-processing property of DP[2].

To be specific, we have used $\tilde{L}_{t-1}^{(0)}$ as the loss clipping threshold R_l for the next iteration in Line 7, because loss values remain similar values within a few iterations; In practice, we set a more conservative loss clipping threshold $R_l = \sum \tilde{L}_{t-1}^{(k)}$ to avoid the clipping bias. We have used the previous η to construct next-loss in Line 6, which in turn will determine the new η by Line 9.

[2]The post-processing of DP ensures that if X is (ϵ, δ)-DP then $g(X)$ is also (ϵ, δ)-DP for any function g.

Algorithm 1 Hyperparameter-free Optimization with Differential Privacy

INPUT: initial η=1e-4, initial $R_l = 1$
Forward pass to compute per-sample losses $L_{t,i}^{(0)} = L(\boldsymbol{w}_t, \boldsymbol{x}_i)$
Compute the mini-batch loss $L_t^{(0)} = \frac{1}{B}\sum_i L_{t,i}^{(0)}$
Back-propagate from $L_t^{(0)}$ to compute $\boldsymbol{m}_{\text{DP}}$ in (2) with Auto Clipping
Post-process $\boldsymbol{m}_{\text{DP}}$ by any optimizer $\mathbf{G}_{\text{DP}} := \mathbf{G}(\boldsymbol{m})$ in (3)
if $t\%K == 0$ (e.g. $K = 10$) **then**
    Forward pass to get per-sample losses $L_{t,i}^{(\pm 1)} = L(\boldsymbol{w}_t \pm \eta \mathbf{G}_{\text{DP}}, \boldsymbol{x}_i)$
    Privatize losses $\tilde{L}_t^{(k)}$ by (7) with $R_l = \tilde{L}_{t-1}^{(0)}$ for $k \in \{-1, 0, +1\}$
    Fit the quadratic function in (6) from $\{-\eta, 0, \eta\}$ to $\{\tilde{L}_t^{(-1)}, \tilde{L}_t^{(0)}, \tilde{L}_t^{(+1)}\}$
    Extract coefficients of the fitted quadratic function $(\boldsymbol{m}^\top \widetilde{\mathbf{H}} \boldsymbol{m})_{\text{DP}}, (\widetilde{\mathbf{G}}^\top \boldsymbol{m})_{\text{DP}}$
    Update η with $\eta_{\text{GeN-DP}} = \frac{(\widetilde{\mathbf{G}}^\top \boldsymbol{m})_{\text{DP}}}{(\boldsymbol{m}^\top \widetilde{\mathbf{H}} \boldsymbol{m})_{\text{DP}}} \approx \text{argmin}_\eta \tilde{L}(\boldsymbol{w}_t - \eta \mathbf{G}_{\text{DP}})$
end if
Update $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \mathbf{G}_{\text{DP}}$

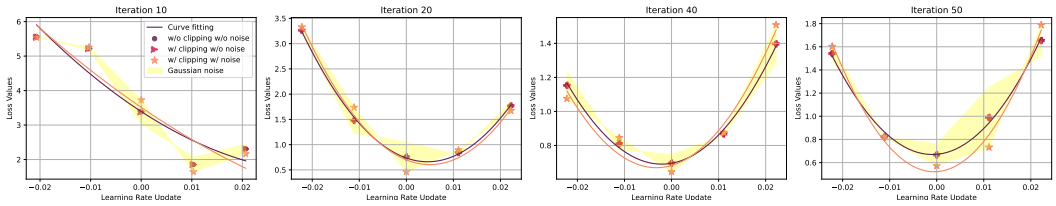

Figure 2: Impact of loss value clipping and perturbation on curve fitting along different training iterations on CIFAR100 with Vit-small fully fine-tuning, with zero in x-axis denotes the current $\boldsymbol{w}_t$. We use 5 points for the ease of illustration and use 3 points in Algorithm 1 and experiments.

4.2 Privacy Guarantee

Theorem 2. *Algorithm 1 is $(\epsilon_{ours}, \delta)$-DP, where ϵ_{ours} depends on the batch size B, the number of iterations T, the noises (σ_g, σ_l), and the update interval K. In contrast, vanilla DP-SGD is $(\epsilon_{vanilla}, \delta)$-DP, where $\epsilon_{vanilla}$ depends on B, T, σ. Furthermore, we have $\epsilon_{ours}(B, T, N, \sigma_g, \sigma_l, K) > \epsilon_{vanilla}(B, T, N, \sigma)$ if $\sigma_g = \sigma$.*

We omit the concrete formulae of ϵ because it depends on the choice of privacy accountants. For example, if we use μ-GDP as an asymptotic estimation, then we show in Appendix B that

$$\mu_{\text{vanilla}} = \sqrt{\left(\frac{B}{N}\right)^2 T(e^{1/\sigma^2} - 1)}, \mu_{\text{ours}} = \sqrt{\mu_{\text{vanilla}}^2 + \left(\frac{B}{N}\right)^2 \frac{3T}{K}(e^{1/\sigma_l^2} - 1)} \tag{9}$$

which can translate into (ϵ, δ)-DP by Equation (6) in Bu et al. (2020). Note in our experiments, we use the improved RDP as the privacy accountant.

Theorem 2 and (9) show that given the same (B, T, σ), our Algorithm 1 uses more privacy budget because we additionally privatize the loss, whereas the vanilla DP-SGD does not protect the hyper-parameter tuning. To maintain the same privacy budget as vanilla DP-SGD, we use an $\approx 1\%$ smaller budget for the gradient privatization, so that the saved budget can be used on the loss privatization. Therefore, we need $\approx 1\%$ larger gradient noise $\sigma_g = \gamma\sigma$ and then select σ_l based on

$$\epsilon_{\text{ours}}(B, T, N, \gamma\sigma, \sigma_l, K) = \epsilon_{\text{vanilla}}(B, T, N, \sigma), \text{where we can use } \gamma \leq 1.01. \tag{10}$$

4.3 END-TO-END NOISE DETERMINATION

We use `autoDP` library[3] to compute σ_l based on σ_g in (10). We give more details of our implementation in Appendix C. We visualize both noises in Figure 3 with RDP accounting and Gaussian and mechanisms for loss privatization, dashed line indicates the case when σ_g and σ_l are set equally. More examples for different mechanisms or different accounting are shown in Appendix. Note that we only adds a little more gradient noise, hence σ_g introduces negligible accuracy drop to Algorithm 1, as empirically shown in Section 5. Additionally, we demonstrate in Figure 2 that σ_l has negligible interference with the precision of estimating η_{GeN}.

4.4 EFFICIENCY OF ALGORITHM

We illustrate that Algorithm 1 can be almost as efficient as the standard non-DP optimization, in terms of training time and memory cost. We identify three orthogonal components that are absent from non-DP optimization: **1) Gradient privatization.** DP optimization (including vanilla DP-SGD) always requires per-sample gradient clipping. Due to the high dimension of gradients, this could incur high cost in memory and time if implemented inefficiently. We directly leverage the recent advances like ghost clipping and book-keeping (BK;) which have allowed DP optimization to be almost as efficient as non-DP optimization, up to 256 GPUs and 100B parameters. **2) Loss privatization.** The cost of loss privatization alone is $O(B)$ and thus negligible, compared to the forward passes and back-propagation which are $O(Bd)$. **3) Learning rate computation.** The cost of computing $\eta_{\text{GeN-DP}}$ in (6) mainly comes from the additional forward passes[4] for $L_{t,i}^{(\pm 1)}$. Given that the back-propagation approximately costs $2\times$ the training time of forward pass, the optimizer without GeN learning rate (non-DP or DP) roughly uses 3 units of time at each iteration. In contrast, Algorithm 1 uses $3 + 2/K \approx 3.2$ units if we set $K = 10$ with $< 7\%$ overhead. We emphasize that the actual overhead to training time is even lower, because the training also includes non-optimization operations such as data loading and inter-GPU communication. In short, the efficiency gap between Algorithm 1 and non-DP optimization is negligible in practice.

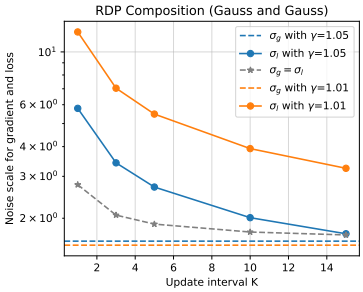

Figure 3: Gradient and loss noise.

5 EXPERIMENTS

To comprehensively evaluate the effectiveness of the proposed method, we conduct experiments on both computer vision tasks and natural language tasks, across different model architectures (Vit Yuan et al. (2021), GPT2 Radford et al. (2019) and Llama2-7B Touvron et al. (2023)) and fine-tuning paradigms (Full, BitFit Zaken et al. (2021) and LoRA Hu et al. (2021)). BitFit only tunes the bias terms of a pre-trained model, while LoRA only tunes the injected low-rank matrices, both keeping all other parameters frozen. Following common practice[5], we set $\delta = N^{-1.1}$ where N is data size.

As the first hyperparameter-free method for differentially private optimization, we compare HyFreeDP with the following baselines: 1) **NonDP-GS**: We manually perform grid search over a predefined range of learning rates, selecting the best without accumulating privacy budget across runs. This serves as the performance upper bound since tuning is non-DP. We also experiment with a manually tuned learning rate scheduler, noted as NonDP-GS w/ LS. We search the learning rate over the range [5e-5, 1e-4, 5e-4, 1e-3, 5e-3] based on previous works Bu et al. (2023b;a) to cover suitable η for various DP levels. 2) **DP-hyper** Liu & Talwar (2019); Wang et al. (2023); Papernot & Steinke (2021): We simulate with a narrow range around the optimal η, spending 85% of the privacy budget for DP training with the searched η. 3) **D-Adaptation** Defazio & Mishchenko (2023) and **Prodigy** Mishchenko & Defazio (2023): Both are state-of-the-art learning rate tuning algorithms in non-DP optimization. We adopt their optimizers with recommended hyperparameters, alongside

[3] `https://github.com/yuxiangw/autodp`

[4] The number of $\{\eta_i\}_i$ in (6) is at least two since there are two unknown variables. More η_i may stabilize the algorithm, at cost of more forward passes, longer training time, and using more privacy budget.

[5] `https://github.com/lxuechen/private-transformers`

Table 2: Performance comparison of HyFreeDP to other baselines. We use consine learning rate decay for NonDP-GS w/ LS baseline in the BitFit fine-tuning. We omit results of $\epsilon = 8$ in Appendix.

Full Fine-Tune		Vit-small					Vit-base				
Privacy budget	Method	CIFAR10	CIFAR100	SVHN	GTSRB	Food101	CIFAR10	CIFAR100	SVHN	GTSRB	Food101
ϵ =1	NonDP-GS	96.48	77.40	90.09	66.96	70.51	95.18	67.62	89.07	80.95	58.76
	D-adaptation	23.94	0.81	15.36	1.86	1.18	19.58	0.83	18.60	4.74	1.48
	Prodigy	28.16	0.81	15.36	1.86	1.19	19.58	0.83	18.60	4.74	1.43
	DP-hyper	93.11	74.64	35.20	28.80	19.40	94.84	6.82	80.56	78.04	56.94
	HyFreeDP	96.49	81.26	93.29	75.51	74.09	96.98	80.94	94.07	84.77	73.07
ϵ =3	NonDP-GS	96.90	83.55	94.41	83.78	74.84	95.80	78.93	91.09	91.05	66.86
	D-adaptation	41.95	0.95	17.30	2.68	1.38	30.58	1.08	19.82	5.68	1.68
	Prodigy	79.06	0.96	18.26	2.73	1.50	30.58	1.08	19.82	5.68	1.67
	DP-hyper	95.11	78.98	49.83	42.52	37.04	95.80	20.64	87.99	90.67	66.13
	HyFreeDP	96.92	84.07	94.44	88.87	78.38	97.52	85.93	94.70	91.48	79.15
NonDP	NonDP-GS	98.34	89.21	95.42	98.64	85.32	98.55	92.37	96.75	98.39	89.53
	D-adaptation	54.90	86.41	97.06	97.65	75.79	63.20	88.36	97.04	98.50	79.47
	Prodigy	97.82	89.26	96.95	98.17	87.45	98.64	90.91	97.03	98.78	89.40
	HyFreeDP	98.23	90.86	96.80	99.00	86.39	98.82	92.33	95.00	94.07	88.70
BitFit Fine-Tune		Vit-small					Vit-base				
Privacy budget	Method	CIFAR10	CIFAR100	SVHN	GTSRB	Food101	CIFAR10	CIFAR100	SVHN	GTSRB	Food101
ϵ =1	NonDP-GS	96.74	84.22	90.18	86.20	77.39	97.34	84.97	90.91	87.15	76.61
	NonDP-GS w/ LS	97.40	81.36	67.95	48.64	74.55	97.90	81.87	79.86	55.66	73.76
	Prodigy	96.36	82.23	90.30	86.85	76.36	97.13	84.87	91.08	81.54	78.72
	HyFreeDP	96.92	86.65	90.02	88.08	82.91	97.55	87.84	92.21	81.83	83.98
ϵ =3	NonDP-GS	97.13	85.95	91.42	91.50	80.37	97.73	87.00	92.20	91.46	80.06
	NonDP-GS w/ LS	97.61	84.87	78.91	60.13	78.08	98.05	85.48	86.22	69.03	78.71
	HyFreeDP	97.23	88.07	89.98	92.38	84.68	97.68	89.53	92.37	90.67	86.26
NonDP	NonDP-GS	97.91	89.39	91.88	95.20	86.29	98.56	91.59	94.42	92.87	88.37
	NonDP-GS w/ LS	97.89	89.40	91.98	90.31	86.27	98.56	91.61	94.43	92.90	88.39
	Prodigy	97.88	87.85	95.19	95.34	86.56	98.41	90.70	96.02	95.02	88.60
	HyFreeDP	97.97	89.86	93.60	95.19	87.24	98.43	91.69	95.48	95.23	89.06

DP-specific clipping and perturbation. 4) **HyFreeDP** : We initialize $R_l = 1$ and $\eta = 1e-4$ by default, allowing automatic updates in training.

5.1 Image Classification Tasks

Experimental setups. As the main result shown in Table 2, we compare HyFreeDP to other baselines by experiments on CIFAR10, CIFAR100, SVHN, GTSRB and Food101 for models of Vit-Small and Vit-Base. We use the privacy budget $\epsilon = \{1, 3, 8\}$ with $\delta \ll n^{-1.1}$ for training dataset with n samples and also perform non-DP baseline ($\epsilon = \infty$), with more setup details in Appendix.

Evaluation results. HyFreeDP outperforms all end-to-end DP baselines. While Non-DP automatic learning rate schedulers (D-adaptation and Prodigy) sometimes match grid-searched constants, they perform poorly in DP training, especially with tighter privacy budgets, confirming our analysis in Section 2.3. Although DP-hyper surpasses these schedulers—likely due to our intentionally narrow search range—finding suitable ranges remains challenging as training dynamics vary by dataset and privacy budget. Even in this optimistic setting, DP-hyper underperforms due to increased gradient noise. In contrast, HyFreeDP achieves consistent performance across various ϵ values and datasets without manual tuning, thanks to our gradient noise control and efficient learning rate estimation with minimal loss value perturbation.

HyFreeDP achieves comparable or superior performance to NonDP-GS baseline without manual learning rate tuning in BitFit experiments. NonDP-GS w/ LS and Prodigy show improved stability compared to full fine-tuning, suggesting sensitivity to training paradigms and trainable model size. In Figure 4, we illustrate training dynamics (R_l, η, loss, test accuracy) across methods. HyFreeDP automatically finds optimal learning rate schedules, with clipping thresholds peaking early and decreasing gradually, enabling more accurate rate estimation as training progresses. While updating with $K = 1$ yields optimal convergence, HyFreeDP remains robust to less frequent updates (e.g., $K = 5$), balancing tuning cost and convergence speed.

5.2 Natural Language Generation Tasks

Experimental setups. We conduct experiments on E2E dataset with a table to text task on GPT-2 model, and also evaluate the language generation task with PubMed dataset by fine-tuning llama2-7B model with LoRA Hu et al. (2021) for demonstrating the scalability and generality of HyFreeDP . We follow the experimental setups based on previous works Bu et al. (2024) and use the dataset provided by Yu et al. (2022; 2023). Based on the Non-DP experience, LoRA typically requires a

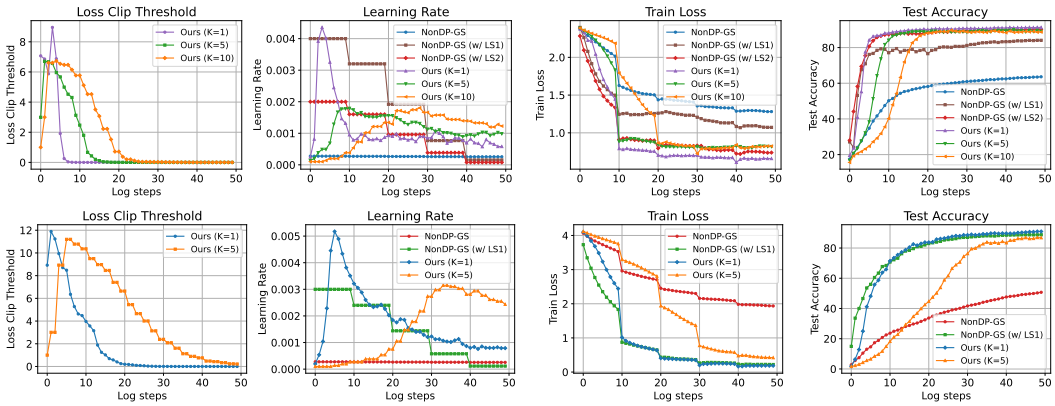

Figure 4: Automatic learning of clipping threshold, learning rate, training loss, and testing accuracy for SVHN (top) and GTSRB (bottom). HyFreeDP schedules R_l and η during training, approaching the manually tuned baseline with end-to-end DP guarantees, and is robust to varying intervals K.

magnitude greater learning rate than fully-finetuning[6], thus we scale up our default initial learning by $\times 10$. We tune the best learning rate for llama2-7B with LoRA fine-tuning on 4,000 samples of PubMed for NonDP-GS when training on the full dataset.

Table 3: Performance comparison on GPT-2 for E2E dataset with different privacy budgets. Best end-to-end DP results are bolded, and results surpassing the manually tuned baseline are underlined.

Full Fine-Tune		$\epsilon = 3$					$\epsilon = 8$				
Model	Method	BLEU	CIDEr	METEOR	NIST	ROUGE_L	BLEU	CIDEr	METEOR	NIST	ROUGE_L
GPT-2	NonDP-GS	0.583	1.566	0.367	5.656	0.653	0.612	1.764	0.385	6.772	0.664
	D-Adaptation	0.000	0.000	0.003	0.082	0.016	0.000	0.000	0.000	0.000	0.000
	Prodigy	0.082	0.000	0.157	1.307	0.239	0.012	0.000	0.003	0.000	0.003
	HyFreeDP	**0.585**	**1.564**	**0.365**	**5.736**	**0.636**	**0.612**	**1.768**	**0.378**	**6.702**	**0.655**

Evaluation results. As shown in Table 3, we observe that even when the privacy budget is not small (e.g., $\epsilon = 8$), Non-DP automatic learning rate scheduler does not perform well. We find that HyFreeDP consistently obtains a comparable performance as the NonDP-GS baseline without extra tuning. In Figure 5, we observe that HyFreeDP automatically discovers a learning rate schedule that achieves better generalization performance compared to the early-stopped NonDP-GS. The automatically determined learning rate (η) reveals a consistent pattern across model scales: an "increase-then-decrease" trajectory that holds true from smaller models (ViT-small) to larger models (Llama2-7B).

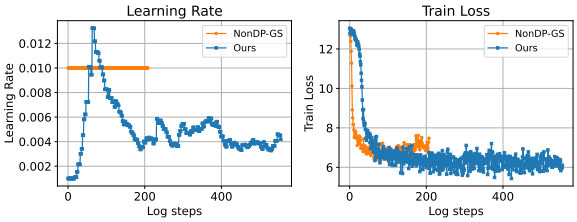

Figure 5: Training dynamics on Llama2-7B.

Table 4: Comparison of model performance in minutes (mins) with and without auto-tuning across various datasets. The coefficients represent the ratio relative to the w/o auto configuration.

Models	Dataset	K=1	K=5	K=10	w/o auto
llama2-7B (LoRA-FT)	PubMed (4k)	409.750 mins ($\times$ 2.040)	244.333 mins ($\times$ 1.217)	222.583 mins ($\times$ 1.108)	200.833 mins ($\times$ 1.000)
GPT2	E2E	163.333 mins ($\times$ 1.888)	97.167 mins ($\times$ 1.123)	94.983 mins ($\times$ 1.098)	86.500 mins ($\times$ 1.000)
Vit-base	CIFAR100	152.617 mins ($\times$ 1.370)	118.483 mins ($\times$ 1.063)	113.317 mins ($\times$ 1.017)	111.433 mins ($\times$ 1.000)
Vit-base (BitFit-FT)	CIFAR100	113.450 mins ($\times$ 1.654)	74.733 mins ($\times$ 1.089)	73.817 mins ($\times$ 1.076)	68.600 mins ($\times$ 1.000)
Vit-small	SVHN	102.000 mins ($\times$ 1.255)	84.500 mins ($\times$ 1.040)	82.800 mins ($\times$ 1.019)	81.250 mins ($\times$ 1.000)

[6] https://docs.anyscale.com/llms/finetuning/guides/lora_vs_full_param/

5.3 Efficiency Comparison

Based on Section 4.4, we compare the training efficiency of HyFreeDP with a single run of DP training using the same NonDP and data-independent hyperparameters, as shown in Table 4. For Llama2-7B, we sample 4,000 records from PubMed and train for 3 epochs on a single A100 (80GB). For smaller datasets and models, we use previous setups on Titan RTX (24GB). HyFreeDP introduces less than $\times 2$ overhead compared to a single run of DP training, even with frequent updates at $K = 1$. Smaller models or those using LoRA or BitFit have lower additional costs, especially with $K = 1$, and the gap narrows as K increases, approaching a cost factor of $1\times$.

6 Discussion and conclusion

In conclusion, we tackle the challenge of hyperparameter tuning in differential privacy (DP) by introducing a hyperparameter-free DP training method that privately and automatically updates the learning rate. Combined with automatic clipping, our approach reduces tuning efforts and ensures end-to-end DP during training. This bridges the gap between hyperparameter-free methods in non-DP settings and DP optimization, opening promising avenues for future research.

References

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

Galen Andrew, Om Thakkar, Brendan McMahan, and Swaroop Ramaswamy. Differentially private learning with adaptive clipping. *Advances in Neural Information Processing Systems*, 34:17455–17466, 2021.

Sourav Biswas, Yihe Dong, Gautam Kamath, and JU COINPRESS. Practical private mean and covariance estimation. *Preprint. Available at*, 2020.

Zhiqi Bu and Shiyun Xu. Automatic gradient descent with generalized newton's method. *arXiv preprint arXiv:2407.02772*, 2024.

Zhiqi Bu, Jinshuo Dong, Qi Long, and Weijie J Su. Deep learning with gaussian differential privacy. *Harvard data science review*, 2020(23), 2020.

Zhiqi Bu, Ruixuan Liu, Yu-Xiang Wang, Sheng Zha, and George Karypis. On the accuracy and efficiency of group-wise clipping in differentially private optimization. *arXiv preprint arXiv:2310.19215*, 2023a.

Zhiqi Bu, Yu-Xiang Wang, Sheng Zha, and George Karypis. Automatic clipping: Differentially private deep learning made easier and stronger. *Advances in Neural Information Processing Systems*, 36, 2023b.

Zhiqi Bu, Yu-Xiang Wang, Sheng Zha, and George Karypis. Differentially private bias-term fine-tuning of foundation models. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=fqeANcjBMT.

Soham De, Leonard Berrada, Jamie Hayes, Samuel L Smith, and Borja Balle. Unlocking high-accuracy differentially private image classification through scale. *arXiv preprint arXiv:2204.13650*, 2022.

Aaron Defazio and Konstantin Mishchenko. Learning-rate-free learning by d-adaptation. In *International Conference on Machine Learning*, pp. 7449–7479. PMLR, 2023.

Jinshuo Dong, Aaron Roth, and Weijie J Su. Gaussian differential privacy. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 84(1):3–37, 2022.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Maor Ivgi, Oliver Hinder, and Yair Carmon. Dog is sgd's best friend: A parameter-free dynamic step size schedule. In *International Conference on Machine Learning*, pp. 14465–14499. PMLR, 2023.

Gautam Kamath, Vikrant Singhal, and Jonathan Ullman. Private mean estimation of heavy-tailed distributions. In *Conference on Learning Theory*, pp. 2204–2235. PMLR, 2020.

Ahmed Khaled, Konstantin Mishchenko, and Chi Jin. Dowg unleashed: An efficient universal parameter-free gradient descent method. *Advances in Neural Information Processing Systems*, 36:6748–6769, 2023.

Itai Kreisler, Maor Ivgi, Oliver Hinder, and Yair Carmon. Accelerated parameter-free stochastic optimization. *arXiv preprint arXiv:2404.00666*, 2024.

Alexey Kurakin, Shuang Song, Steve Chien, Roxana Geambasu, Andreas Terzis, and Abhradeep Thakurta. Toward training at imagenet scale with differential privacy. *arXiv preprint arXiv:2201.12328*, 2022.

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

Jingcheng Liu and Kunal Talwar. Private selection from private candidates. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pp. 298–309, 2019.

Zhihao Liu, Jian Lou, Wenjie Bao, Zhan Qin, and Kui Ren. Differentially private zeroth-order methods for scalable large language model finetuning. *arXiv preprint arXiv:2402.07818*, 2024.

Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D Lee, Danqi Chen, and Sanjeev Arora. Fine-tuning language models with just forward passes. *Advances in Neural Information Processing Systems*, 36:53038–53075, 2023.

Konstantin Mishchenko and Aaron Defazio. Prodigy: An expeditiously adaptive parameter-free learner. *arXiv preprint arXiv:2306.06101*, 2023.

Shubhankar Mohapatra, Sajin Sasy, Xi He, Gautam Kamath, and Om Thakkar. The role of adaptive optimizers for honest private hyperparameter selection. In *Proceedings of the aaai conference on artificial intelligence*, volume 36, pp. 7806–7813, 2022.

Nicolas Papernot and Thomas Steinke. Hyperparameter tuning with renyi differential privacy. *arXiv preprint arXiv:2110.03620*, 2021.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

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

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Hua Wang, Sheng Gao, Huanyu Zhang, Weijie J Su, and Milan Shen. Dp-hypo: an adaptive private hyperparameter optimization framework. *arXiv preprint arXiv:2306.05734*, 2023.

Xiaodong Yang, Huishuai Zhang, Wei Chen, and Tie-Yan Liu. Normalized/clipped sgd with perturbation for differentially private non-convex optimization. *arXiv preprint arXiv:2206.13033*, 2022.

Da Yu, Saurabh Naik, Arturs Backurs, Sivakanth Gopi, Huseyin A Inan, Gautam Kamath, Janardhan Kulkarni, Yin Tat Lee, Andre Manoel, Lukas Wutschitz, et al. Differentially private fine-tuning of language models. In *International Conference on Learning Representations (ICLR)*, 2022.

Da Yu, Arturs Backurs, Sivakanth Gopi, Huseyin Inan, Janardhan Kulkarni, Zinan Lin, Chulin Xie, Huishuai Zhang, and Wanrong Zhang. Training private and efficient language models with synthetic data from llms. In *Socially Responsible Language Modelling Research*, 2023.

Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 558–567, 2021.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*, 2021.

Liang Zhang, Kiran Koshy Thekumparampil, Sewoong Oh, and Niao He. Dpzero: Dimension-independent and differentially private zeroth-order optimization. In *International Workshop on Federated Learning in the Age of Foundation Models in Conjunction with NeurIPS 2023*.

A Experimental Details

We also fix other hyperparameters not explicitly displayed in Algorithm 1, e.g. throughout this paper, we fix the momentum coefficients and weight decay in AdamW at $(\beta_1, \beta_2, \text{weight\_decay}) = (0.9, 0.999, 0.01)$, which is the default in Pytorch.

Image classification. We tuned the learning rate for non-automatic baselines according to the recommended learning rate range provided in previous work Bu et al. (2023b). In fully fine-tuning, we tried to integrate a constant linear scheduler as a common practice, but the result does not outperform the tuned NonDP-GS, so we omit the results in Appendix Table 5. We do not apply the linear scheduler to Prodigy and D-adaptation for keeping the originally recommended configuration. The results with $\epsilon = 8$ in Table 2 is omitted in Table 6 with the consistent conclusions across different datasets.

	Vit-small					Vit-base				
Privacy Budget	CIFAR10	CIFAR100	SVHN	GTSRB	Food101	CIFAR10	CIFAR100	SVHN	GTSRB	Food101
$\epsilon = 1$	88.02	3.46	29.00	10.16	9.80	96.62	58.74	89.13	64.39	60.74
$\epsilon = 3$	92.48	8.00	35.54	17.68	20.40	97.11	75.73	91.79	82.31	69.09
$\epsilon = 8$	93.79	15.04	43.73	24.15	31.05	97.41	81.31	93.05	88.79	73.65
NoDP	98.00	89.11	96.55	96.94	84.55	98.88	92.51	97.27	98.29	89.10

Table 5: Fully fine-tuning results by adding a linear scheduler to NonDP-GS across different datasets with privacy budgets for Vit-small and Vit-base models. Results show that directly integrating a constant learning rate scheduler in NonDP does not hurt performance but the DP training performance is sensitive to the learning rate scheduler.

Additionally, we demonstrate the loss clipping bias in Figure 7.

Table 6: Performance comparison of HyFreeDP to other baselines. We use consine learning rate decay for NonDP-GS w/ LS baseline in the BitFit fine-tuning setting.

Full Fine-Tune		Vit-small					Vit-base				
Privacy budget	Method	CIFAR10	CIFAR100	SVHN	GTSRB	Food101	CIFAR10	CIFAR100	SVHN	GTSRB	Food101
ϵ =8	NonDP-GS	96.28	84.99	92.53	89.97	77.08	96.14	82.52	92.38	94.91	71.05
	D-adaptation	78.31	1.07	19.10	3.67	1.76	40.90	1.25	20.86	6.84	1.94
	Prodigy	95.74	1.29	20.75	4.86	2.92	45.89	1.25	20.86	6.84	1.90
	DP-hyper	95.70	80.58	64.72	50.10	49.18	96.28	36.46	90.27	94.62	70.63
	HyFreeDP	97.04	86.00	95.15	88.06	80.61	97.79	87.57	95.00	94.07	81.91
BitFit Fine-Tune		Vit-small					Vit-base				
Privacy budget	Method	CIFAR10	CIFAR100	SVHN	GTSRB	Food101	CIFAR10	CIFAR100	SVHN	GTSRB	Food101
ϵ =8	NonDP-GS	97.23	86.56	92.23	93.34	82.12	97.81	88.05	93.38	93.04	81.88
	NonDP-GS w/ LS	97.63	86.21	83.17	67.35	79.85	98.08	87.17	88.25	75.61	81.18
	HyFreeDP	97.31	88.43	89.17	93.54	74.61	97.90	90.34	92.36	93.14	87.14

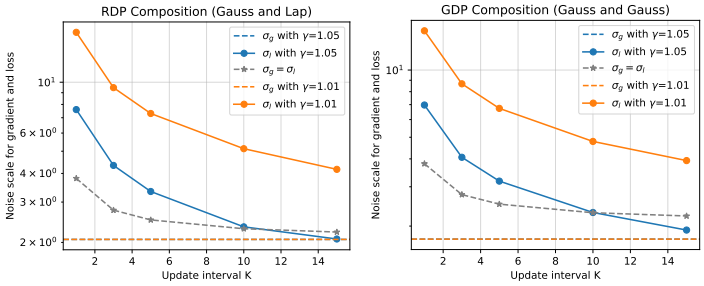

Figure 6: Privacy composition for gradient and loss values privatization, with privacy accountants of RDP and GDP, and loss perturbation of Gaussian and Laplacian mechanisms. Gray dashed line indicates the naive and even budget splitting for every access to private gradient and loss, which results in larger noise magnitude on gradient especially when the adjustment is frequent. The privacy accounting strategy proposed in HyFreeDP effectively restrain the privacy budget consumption on hyper-parameter tuning and spending it wisely by only perturbing a single-dimensional loss value.

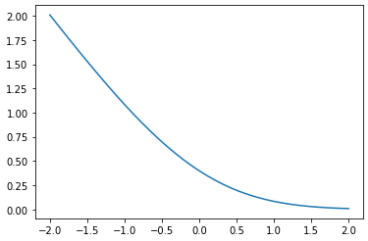

Figure 7: $\phi(\alpha) - \alpha(1 - \Phi(\alpha))$ which is minimized at $\alpha = \infty$.

B PROOFS

Proof of Theorem 1. It's not hard to see

$$\mathbb{E}(\tilde{L}) - \frac{\sum_i L_i}{B} = \frac{1}{B}\sum_i \min(\frac{R_l}{L_i}, 1)L_i - \frac{1}{B}\sum_i L_i$$
$$= \frac{1}{B}\sum_i (R_l - L_i)\mathbb{I}(L_i > R_l) = -\frac{1}{B}\sum_i \text{ReLU}(L_i - R_l)$$

in which the non-negative $\text{ReLU}(x) = x \cdot \mathbb{I}(x > 0)$. Taking the absolute value, we have

$$\left|\mathbb{E}(\tilde{L}) - \frac{\sum_i L_i}{B}\right| = \frac{1}{B}\sum_i \text{ReLU}(L_i - R_l) = \left|\frac{1}{B}\sum_i (L_i - R_l)\mathbb{I}(L_i > R_l)\right|$$

which is decreasing in R_l because ReLU is increasing in its input $(L_i - R_l)$, and this input is decreasing in R_l.

As $B \to \infty$, the clipping bias tends to

$$\mathbb{E}(\text{ReLU}(L_i - R_l)) = \mathbb{E}(\text{ReLU}(L_i - R_l)|L_i > R_l) \cdot \mathbb{P}(L_i > R_l)$$
$$= \mathbb{E}((L_i - R_l)|L_i > R_l) \cdot \mathbb{P}(L_i > R_l) = [\mathbb{E}(L_i|L_i > R_l) - R_l] \cdot \mathbb{P}(L_i > R_l)$$

where we have used $\text{ReLU}(x) = x$ when $x > 0$. □

Proof of Corollary 1. The key part in (8) is $\mathbb{E}(L_i|L_i > R_l)$, which is the expectation of the truncated normal distribution by one-side truncation. It is known that for $\alpha = \frac{R_l - \mu}{\xi}$,

$$\mathbb{E}(L_i|L_i > R_l) = \mu + \xi\frac{\phi(\alpha)}{1 - \Phi(\alpha)}, \quad \mathbb{P}(L_i > R_l) = 1 - \Phi(\alpha)$$

The proof is complete by inserting these quantities. □

Proof of Theorem 2. All privacy budget of Algorithm 1 goes into two components: privatizing the gradient (with noise level σ_g) and privatizing the loss (with noise level σ_l).

Under the same (B, T, N, σ_g), we have T mechanisms of gradient privatization, each of (ϵ_g, δ_g)-DP and $3T/K$ mechanisms of loss privatization, each of (ϵ_l, δ_l)-DP. Hence it is clear that $\epsilon_{\text{ours}} > \epsilon_{\text{vanilla}}$.

To be more specific, we demonstrate with μ-GDP. The vanilla DP-SGD is μ-GDP with

$$\mu_{\text{vanilla}} = \frac{B}{N}\sqrt{T(e^{1/\sigma_g^2} - 1)}$$

which is the same as the gradient privatization component of our DP-SGD. We additionally spend

$$\mu_l = \frac{B}{N}\sqrt{\frac{3T}{K}(e^{1/\sigma_l^2} - 1)}$$

leading to a total budget of

$$\mu_{\text{ours}} = \sqrt{\mu_{\text{vanilla}}^2 + \mu_l^2}$$

by Corollary 3.3 in Dong et al. (2022). It is clear $\mu_{\text{ours}} > \mu_{\text{vanilla}}$. □

C END-TO-END PRIVACY ACCOUNTING AND INVERSE

We demonstrate how to determine (σ_g, σ_l) given (ϵ, δ)-DP budget. In vanilla DP optimization, we can leverage privacy accountants such as RDP, GDP, PRV, etc. Each accountant is a function whose input is hyperparameters $(B, T, N, \delta, \sigma)$ and the output is ϵ (see an example in (9)). In this section, we denote any accountant as f, so that

$$f(\sigma; B, N, \delta) = \epsilon' \tag{11}$$

$$f^T(\sigma; B, N, \delta) = \epsilon \tag{12}$$

$$f^{-T}(\epsilon; B, N, \delta) = \sigma \tag{13}$$

in which ϵ' is the single-iteration budget, f^T means a composition of T iterations, and f^{-T} is the inverse function known as `GetSigma` in Figure 1.

In this work, we develop an end-to-end privacy accountant to compose both the gradient privatization and the loss privatization. Our accountant takes the input $(B, T, N, \delta, \sigma_l, \gamma, K)$, where $\gamma = 1.01$ by default and can take smaller value for larger models or smaller (ϵ, δ) budget. Therefore, $\sigma_g = \gamma\sigma$.

Firstly, we call $f^T(\gamma\sigma; B, N, \delta) = \hat{\epsilon}$ to get the reference budget $\hat{\epsilon}$ which is strictly smaller than ϵ because f is monotonically decreasing in its input. Then we guess the loss noise and call $f^{3T/K}(\sigma_l; B, N, \delta) = \epsilon_l$ since there are $3T/K$ rounds of loss privatization. We continue our guess until

$$f^{3T/K}(\sigma_l; B, N, \delta) + f^T(\gamma\sigma; B, N, \delta) = \epsilon_l + \hat{\epsilon} = \epsilon$$

Notice the left hand side is monotonically decreasing in σ_l. Hence we use bisection method to find the unique solution σ_l, at an exponentially fast speed.

Algorithm 2 End-to-End Privacy Accounting

INPUT: The end-to-end DP budget (ϵ, δ), `GetSigma`$(\cdot)$, `Compose`$(\cdot)$, `Solve`$(\cdot)$
OUTPUT: Noise magnitude for gradient and loss privatization σ_g and σ_l
▷ *Compute the gradient noise scale σ by assuming there is only a single training run*
$\sigma =$ `GetSigma`$(\epsilon, \delta, B, T, N)$
▷ *Compute the σ_g in Algorithm 1 with a controlled noise increase*
$\sigma_g = \gamma \cdot \sigma$ with the constant γ slightly greater than 1 (e.g, $\gamma = 1.01$)
▷ *Define the composition function with input variable as the loss noise scale c*
$\epsilon_{\text{ours}}(c|\sigma_g, \delta, B, N, T, K) =$ `Compose`$(f_g^T(\sigma_g; B, N, \delta), f_l^{3T/K}(c; B, N, \delta))$
▷ *Solve the minimization of the scalar function respect to c*
$\sigma_l =$ `Solve`$(c, \epsilon) = \arg\min_c |\epsilon_{\text{ours}}(c) - \epsilon|$

In the above, we have used the functions `GetSigma`$(\cdot)$[7], `Compose`$(\cdot)$[8], and any root-finding method `Solve`$(\cdot)$ such as the bi-section in `scipy` library. We highlight that Algorithm 1 and Algorithm 2 are sufficiently flexible to work with the general DP notions, including GDP, Renyi DP, tCDP, per-instance DP, per-user DP, etc.

D MISC

Hyper-parameter matters for DP optimization. Previous works De et al. (2022); Li et al. (2021) reveal that the performance of DP optimization is sensitive to the hyper-parameter choices, as we cited in Figure 8. On the one hand, DP by itself brings extra hyper-parameters, such as the gradient clipping threshold R_g, making the tuning more complex. An adaptive clipping method Andrew et al. (2021) proposes to automatically learn R_g at each iteration, with an extra privacy budget that translates to worse accuracy. While there are methods that does not incur additional privacy budget. Automatic clipping (or per-sample normalization, or AutoClip) Bu et al. (2023b); Yang et al. (2022) is a technique that uses $c_i = 1/||\boldsymbol{g}_i||$ to replace the R_g-dependent per-sample clipping, and thus removes the hyperparameter R_g from DP algorithms.

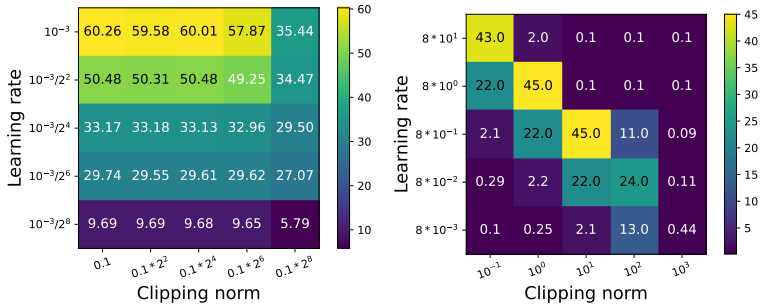

Figure 8: Hyperparameter tuning of (R_g, η) cited from Figure 1 of Bu et al. (2023b). Here R_g is the clipping norm and η is the learning rate. Left: BLEU score of GPT2 on E2E dataset Li et al. (2021). Right: Test accuracy of ResNet18 on ImageNet Kurakin et al. (2022).

End-to-end DP guarantee for optimization and tuning. As shown in Table 7, we compare our work with other works that try to ensure end-to-end DP guarantee for both optimization and tuning. As we categorized, there are two orthogonal solutions.

1) One line of works focus on DP-hyper, which guarantee DP for the whole process of hyper-parameter tuning and training Mohapatra et al. (2022); Papernot & Steinke (2021); Wang et al. (2023). Recently, Liu & Talwar (2019) enhanced DP bounds for hyper-parameter tuning and showed that the searching process repeated a random number of times satisfies $(3\epsilon, 0)$-DP if each run was $(\epsilon, 0)$-DP. Papernot & Steinke (2021) extends by offering more precise Rényi DP guarantees. Wang et al. (2023) proposes an adaptive hyper-parameter search based on previous results. In summary, these works either require a given prior knowledge for efficient searching or consumes a large portion of privacy budget for effective searching. We cited with the representative work of Papernot & Steinke (2021) in Table 7.

2) The other orthogonal line of works make DP optimization hyper-parameter free, such as Auto Clipping Bu et al. (2023b), or spend a portion of budget during training to tune the clipping threshold as the Adaptive Clipping Andrew et al. (2021). We are positioned in the second line of works and we target to make learning rate hyper-parameter free.

[7] `https://github.com/yuxiangw/autodp/blob/master/example/example_calibrator.py`

[8] `https://github.com/yuxiangw/autodp/blob/master/example/example_composition.py`

Table 7: Comparison of hyperparameter search strategies. Budget splitting percentages for Adaptive Clipping and DP-hyper are estimated values, as the actual percentages depend on specific datasets.

Method	Searching η	Searching R_g	% Budget on Hyperparam
Vanilla Abadi et al. (2016)	✓	✓	0%
Auto Clipping Bu et al. (2023b)	✓	×	0%
Adaptive Clipping Andrew et al. (2021)	✓	×	≈20%
DP-hyper Papernot & Steinke (2021)	✓	×	≈20%
Ours (this work)	×	×	<1%