Revisiting Differentially Private Hyper-parameter Tuning

Zihang Xiang
KAUST
zihang.xiang@kaust.edu.sa

Tianhao Wang
University of Virginia
tianhao@virginia.edu

Cheng-Long Wang
KAUST
chenglong.wang@kaust.edu.sa

Di Wang*
KAUST
di.wang@kaust.edu.sa

***Abstract*—We investigate the application of differential privacy in hyper-parameter tuning, a process involving selecting the best run from several candidates. Unlike many private learning algorithms, including the prevalent DP-SGD, the privacy implications of selecting the best are often overlooked. While recent works propose a generic *private selection* solution for the tuning process, an open question persists: is such privacy upper bound tight?**

This paper provides both empirical and theoretical examinations of this question. Initially, we provide studies affirming the current privacy analysis for private selection is indeed tight in general. However, when we specifically study the hyper-parameter tuning problem in a white-box setting, such tightness no longer holds. This is first demonstrated by applying privacy audit on the tuning process. Our findings underscore a substantial gap between the current theoretical privacy bound and the empirical privacy leakage derived even under strong audit setups.

This gap motivates our subsequent theoretical investigations, which provide improved privacy upper bound for private hyper-parameter tuning due to its distinct properties. Our improved bound leads to better utility. Our analysis also demonstrates broader applicability compared to prior analyses, which are limited to specific parameter configurations. Overall, we contribute to a better understanding of how privacy degrades due to *selection*.

I. Introduction

Differential Privacy (DP) [18] stands as the prevailing standard for ensuring privacy in contemporary machine learning. A ubiquitous technique employed to ensure DP across a diverse array of machine learning tasks is differentially private stochastic gradient descent (DP-SGD, *a.k.a.*, noisy-SGD) [5], [45], [2].

In addition to a single (private) training process, machine learning systems always involve a hyper-parameter tuning process that entails running a (private) *base* algorithm (e.g., DP-SGD) multiple times with different configurations and selecting the best run. Regrettably, unlike the well-studied DP-SGD, the reasoning for the privacy cost of such tuning operations is inadequately studied and often totally ignored.

*Corresponding author

Network and Distributed System Security (NDSS) Symposium 2026
23-27 February 2026, San Diego, CA, USA
ISBN 979-8-9919276-8-0
https://dx.doi.org/10.14722/ndss.2026.230447
www.ndss-symposium.org

Naively, one can bound the privacy loss for the tuning operation by the composition theorem. If we run the private base algorithm k times with different hyper-parameters, the total privacy cost deteriorates at most linearly with k (or $\mathcal{O}(\sqrt{k})$ if the base algorithm is approximate DP and we use advanced composition theorem [20], which is nearly optimal [24]). However, these bounds are still far from satisfactory as k is usually large in practice. Perhaps due to this limitation, it still remains common to exhaustively tune a private algorithm to achieve strong performance but only consider the privacy cost for a single run [14], [59], [54], [43], [56], [55].

For another approach, tuning hyper-parameter privately can be framed as a *private selection* problem, for which several well-explored mechanisms, such as the sparse vector technique [19] and the exponential mechanism [32], may potentially be utilized. However, these mechanisms assume that the score function (defining the "best" to be selected) has low sensitivity (for DP analysis), which is a condition not always met.

Thanks to Liu and Talwar [30], hyper-parameter tuning now enjoys significantly better privacy bound than naively applying composition theorem. To briefly describe their findings, if we run a private base algorithm a *random* number of times (possibly with different hyper-parameters) and only output the best single run, the privacy cost only deteriorates by a constant multiplicative factor [30]. For example, if the base algorithm is $(\varepsilon, 0)$-DP, then the whole tuning process is $(3\varepsilon, 0)$-DP if the running number follows a geometric distribution [30]. This is much better than the $(k\varepsilon, 0)$-DP bound under *fixed* k times of running.

Later, Papernot and Steinke [42] operate within Rényi DP (RDP) framework [33] and mandate the randomization of the number of running times, presenting additional results for varying degrees of randomness. A noteworthy aspect of both methodologies [30], [42] lies in their treatment of the base algorithm as a *black box*; thus, such a generic approach applies to a broader spectrum of private selection problems, provided the base algorithm is differentially private on its own.

Motivations. Hyper-parameters can be tuned with formal privacy guarantees. But it remains unclear whether the existing privacy analyses [30], [42] are tight.

Some results suggest the cost of tuning is high. For example, [30] shows the privacy budget can increase by a factor of three in certain setups. Still, this seems counterintuitive. It seems plausible that *only* revealing the best *single* run should

not consume that much privacy budget. This leads to our core question: Does hyperparameter tuning actually consume significantly more privacy than the base algorithm?

If the answer is positive, then further significant improvements in the analysis is impossible. If negative, it is still valuable to pursue tighter analysis, as tighter bounds would allow more hyperparameter trials under the same budget, directly improving utility in private model selection.

This work. We answer the posed question with both positive and negative answers. In the affirmative, our constructed example demonstrates that the current generic privacy bound provided in [42] for private selection is indeed tight. Still, the result only holds in the worst case. Conversely, in the negative, we uncover a more favorable privacy bound given the base algorithm is specific DP-SGD. We aim to understand how *selection* leaks privacy, in contrast to the well-established understanding of privacy deterioration due to *composition*. Our contributions are as follows.

1) Validating tightness of generic privacy bound for private selection (Section III-B). We first provide a private selection instance where we observe only a negligible gap between the true privacy cost and the cost predicted by the current privacy bound [42]. Such results meaningfully show that significant improvement in privacy upper bound *is impossible in general*. However, when we study the private hyper-parameter tuning problem, where the base algorithm is DP-SGD, we enjoy better upper bounds. This finding is related to our other two contributions.

2) Empirical investigation on how hyper-parameter tuning (selection) leaks privacy (Section IV). We first take empirical approaches to investigate how much privacy is leaked when performing hyper-parameter tuning. This is done via the privacy audit technique [39], [24], [23], an interactive protocol used to empirically measure the privacy of some mechanisms.

In contrast, unlike all previous privacy auditing work, which focuses on the privacy of the base algorithm (e.g., DP-SGD), auditing the tuning procedure is a fresh problem that requires new formulation and insight. Specifically, the score function used to select the "best" is the new factor that must be settled.

We formulate various privacy threat models tailored for hyper-parameter tuning, where the weakest one corresponds to the most practical scenario and the strongest one corresponds to the worst case. Results under the weakest provide evidence that the tuning process hardly incurs additional privacy costs beyond the base algorithm. Moreover, even the empirical privacy bound (lower bound) derived from the strongest adversary still exhibits a substantial gap from the generic privacy bound (upper bound) proposed by [42]. *In contrast, previously, the gap (between privacy lower bound and upper bound) is essentially closed in the audit on DP-SGD's privacy [39].* Why are different audit results seen in auditing hyper-parameter tuning? This motivates our subsequent theoretical investigations.

3) Improved theoretical privacy results (Sections V-B, V-C and VI). Our subsequent study shows that tuning DP-SGD does enjoy a better privacy result. The pivotal aspect driving this improvement lies in representing the privacy of the base algorithm with finer resolution, and DP-SGD does have a distinctive characterization. This is done within the f-DP framework [17], deviating from the well-known (ε, δ)-DP [18] or RDP [33].

Our improved result directly benefits differentially private hyper-parameter tuning: *it allows us to test substantially more (in expectation) hyper-parameters without increasing privacy budget, which translates to improved utility*. Our results are generalizable, contrasting to previous work [30], [42], which remains unknown how to adapt to general parameter setups.

Subsequent to our improved results is a further empirical evaluation: comparing our improved theoretical privacy result with the empirical privacy lower bound derived under an idealized audit setup. Interestingly, there is still a gap in between. This finding is examined in detail, revealing that the score function, a new factor in auditing hyper-parameter tuning, is a key determinant influencing audit performance. Consequently, this also prompts an exciting and essential open problem in the future: how to close such a gap.

II. Background

A. Differential Privacy (DP)

Definition 1 (Differential Privacy [18])**.** *Given a data universe $\mathcal{X}$, two datasets $X, X' \subseteq \mathcal{X}$ are adjacent if they differ by one data example. A randomized algorithm $\mathcal{M}$ satisfies (ε, δ)-differential privacy, or (ε, δ)-DP, if for all adjacent datasets X, X' and for all events S in the output space of $\mathcal{M}$, we have* $\Pr(\mathcal{M}(X) \in S) \leq e^{\varepsilon} \Pr(\mathcal{M}(X') \in S) + \delta$.

We introduce Rényi DP (RDP), a DP relaxation shown in the following, often serves as a tight analytical tool to assess the privacy cost under composition.

Definition 2 (Rényi DP [34])**.** *The Rényi divergence is defined as $\mathcal{D}_\alpha(M||N) = \frac{1}{\alpha-1} \ln \mathbb{E}_{x\sim N} \left[\frac{M(x)}{N(x)}\right]^\alpha$ with $\alpha > 1$. A randomized mechanism $\mathcal{M} : \mathcal{X} \to \mathcal{Y}$ is said to be (α, γ)-Rényi DP, or (α, γ)-RDP, if $\mathcal{D}_\alpha(\mathcal{M}(X)||\mathcal{M}(X')) \leq \gamma$ holds for any adjacent dataset X, X'.*

Differentially private stochastic gradient descent (DP-SGD) [5], [45], [2]. We use a machine learning model f_w, typically a neural network with trainable parameters w. In our classification setting, f_w maps inputs (e.g., images) to labels. Parameters are updated using Stochastic Gradient Descent (SGD) [28], where hyper-parameters like learning rate must be tuned for good performance.

DP-SGD is the private version of SGD. It follows three steps: 1) compute per-sample gradients; 2) clip each to have bounded ℓ_2 norm; 3) add Gaussian noise.

The private gradient p_i is then used to update w as:

$$\begin{aligned} p_i &= \sum_{(x,y)\in\mathbf{B}} \mathrm{CLP}_C\left(\nabla_w \ell\left(w_{i-1}; x, y\right)\right) + R_i \\ w_i &\leftarrow w_{i-1} - \mathbf{lr} \cdot p_i \end{aligned} \tag{1}$$

Here, $\mathbf{B}$ is the sampled batch (with ratio τ), $\mathbf{lr}$ is the learning rate, and ℓ is the loss (e.g., cross-entropy). Clipping is

defined as $\mathrm{CLP}_C(u) = u \cdot \min(1, \frac{C}{|u|_2})$, where C is a clipping threshold. Noise R_i is sampled from $\mathcal{N}(0, C^2\sigma^2\mathbb{I}^d)$, where σ is the noise multiplier and d is the number of parameters. Removing clipping and noise recovers (mini-batch) SGD. Variants like DP-Adam [48] follow the same privacy analysis by the post-processing property of DP.

B. Privacy Audit

Hypothesis testing interpretation of DP. For a randomized mechanism $\mathcal{M}$, let X, X' be adjacent datasets, let $y \in \mathcal{Y}$ be the output of $\mathcal{M}$ taking input X or X', we form the *null* and *alternative* hypotheses:

$$\mathbf{H_0} : X \text{ was the input}, \quad \mathbf{H_1} : X' \text{ was the input}. \tag{2}$$

For any decision rule $\mathcal{R} : \mathcal{Y} \to \{0, 1\}$ in such a hypothesis testing problem, it has two notable types of errors: 1) type I error or false positive rate $\mathrm{FP} = \Pr(\mathcal{R}(y) = 1|\mathbf{H_0})$, i.e., the probability of rejecting $\mathbf{H_0}$ while $\mathbf{H_0}$ is true; 2) type II error or false negative rate $\mathrm{FN} = \Pr(\mathcal{R}(y) = 0|\mathbf{H_1})$, i.e., the probability of rejecting $\mathbf{H_1}$ while $\mathbf{H_1}$ is true. DP can be characterized by such two error rates as follows.

Theorem 1 (DP as Hypothesis Testing [24])**.** *For any $\varepsilon > 0$ and $\delta \in [0, 1]$, a mechanism $\mathcal{M}$ is (ε, δ)-DP if and only if*

$$\mathrm{FP} + \mathrm{e}^{\varepsilon}\mathrm{FN} \geq 1 - \delta, \quad \mathrm{FN} + \mathrm{e}^{\varepsilon}\mathrm{FP} \geq 1 - \delta \tag{3}$$

both hold for any adjacent dataset X, X' and any decision rule $\mathcal{R}$ in a hypothesis testing problem as defined in Equation (2)*.*

Theorem 1 has the following implications. With δ fixed at some value, under the threat model that an adversary can only operate at some FP and FN under some decision rule $\mathcal{R}$ for a specific adjacent dataset pair X, X', a *lower bound*

$$\varepsilon_L^{(X,X',\mathcal{R})} = \max\{\log \frac{1 - \delta - \mathrm{FP}}{\mathrm{FN}}, \log \frac{1 - \delta - \mathrm{FN}}{\mathrm{FP}}, 0\} \tag{4}$$

can be computed, meaning that the algorithm cannot be more private than that, i.e., the true privacy parameter $\varepsilon_T \geq \varepsilon_L^{(X,X',\mathcal{R})}$, just as entailed by Theorem 1. Finding ε_T requires taking the maximum of lower bound value over all pairs of X, X' and $\mathcal{R}$, which is clearly intractable in general. In practice, people are satisfied by reporting an *upper bound* $\varepsilon_U \geq \varepsilon_T$, which is obtained by analytical approaches (privacy accounting) [2], [33], [35].

Algorithm 1 Game-based Privacy Audit $\mathcal{G}$

Input: DP protocol $\mathcal{P}$, adjacent pair X, X'
1: $b_{\text{truth}} \leftarrow \{0, 1\}$ ▷ *Trainer* flips a fair coin
2: $\hat{X} \leftarrow X$ if $b_{\text{truth}} = 0$, $\hat{X} \leftarrow X'$ otherwise
3: Run $\mathcal{P}(\hat{X})$ ▷ *Trainer* runs the private protocol
4: $b_{\text{guess}} \leftarrow \{0, 1\}$ ▷ *Adversary makes a guess based on* $\mathcal{P}(\hat{X})$
Output: $(b_{\text{truth}}, b_{\text{guess}})$

Privacy audit. Privacy audit aims to find a lower bound of the privacy cost for a private protocol $\mathcal{P}$ based on the hypothesis testing interpretation of DP as shown above. This is usually done via simulating the interactive game-based protocol described in Algorithm 1. Such a simulation is typically repeated many times, resulting in many pairs of $(b_{\text{truth}}, b_{\text{guess}})$. Then, the FP and FN for adversary's guessing are computed by Clopper-Pearson method [10] with a confidence specification. If the adversary can make very accurate guesses and derive a lower bound higher than some claimed privacy parameter, it suggests $\mathcal{P}$ is not private as claimed.

Audit only gives a lower bound $\varepsilon_L^{(X,X',\mathcal{R})}$ of the true privacy bound ε_T, meaning that the algorithm is at least not $(\varepsilon_L^{(X,X',\mathcal{R})}, \delta)$. The lower bound due to privacy audit is different from the upper bound given by theory. The limitation of privacy audit is that the result it gives should not be used as a formal privacy guarantee.

Related work on privacy audit. In privacy-preserving machine learning, privacy audit mainly serves a different goal from that of certain earlier studies [52], [16], [7], [6] on detecting privacy violation in general query-answering applications. Previous work on privacy audit in machine learning mainly targets auditing the DP-SGD protocol to assess its theoretical versus practical privacy [39], [24], [23]. Additional studies [47], [37], [31], [60], [57] concentrate on enhancing the strength of audits on DP-SGD (yielding stronger/larger-value lower bound) or improving the efficiency (incurring fewer simulation overheads). Drawing a parallel to the action of guessing whether a data point was included or not, privacy audit may also be linked to *membership inference attack* (MIA) [38], [44]. Still, privacy audit aims to give a privacy lower bound. There are also recent works on auditing prediction [8] and synthetic data generation [3], which differ from our auditing experiments.

C. Private Hyper-parameter Tuning

Problem Formulation. We formulate the private hyper-parameter tuning problem aligning with [30], [42]. Let $\Omega = \{\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_m\}$ be a collection of DP-SGD algorithms (m is chosen freely). These correspond to m possible hyper-parameter configurations. We have $\mathcal{M}_i : \mathcal{X} \to \mathcal{Y}$ for $i \in [m]$, and all of these algorithms satisfy the same privacy parameter, e.g., they satisfy the same (ε, δ)-DP guarantee.

Note that it requires that each $\mathcal{M}_i$ to be differentially private on its own [30], [42], meaning that indistinguishability exists between distribution $\mathcal{M}_i(X)$ and $\mathcal{M}_i(X')$, for any X, X' being adjacent and for any $\mathcal{M}_i$'s hyper-parameter.

Finally, it is to return an algorithm element (including its execution) of Ω such that the output of such algorithm has (approximately) the maximum score as specified by some score function $g : \mathcal{Y} \to \mathbb{R}$. The score function g usually serves a utility purpose (e.g., g could evaluate the validation loss on a held-out dataset). The selection must be performed in a differentially private manner. The general *private selection* problem corresponds to the cases where Ω contains arbitrary differentially private algorithms.

Related work on private hyper-parameter tuning. Well-known algorithms like the sparse vector technique [19] and exponential mechanism [32] may potentially be leveraged to

the tuning problem; however, they assume a low sensitivity in the metric defining the "best", a condition not always applicable. Some earlier work [9] also suffers from the same issue. Papernot et al. [42] and Liu and Talwar [30] have provided generic private selection approaches circumventing such challenges. Mohapatra et al. [36] study privacy issues in *adaptive* hyper-parameter tuning under DP, which is different from the *non-adaptive* tuning problem considered in this work. There is related work [25] that integrates the solution from [42] to some larger algorithm; therefore, what we understand about the generic approach in this study naturally propagates to [25].

Focus of this paper. Towards understanding how selection leaks privacy, our first focus is to formulate specific privacy audit to understand how privacy deteriorates due to *selection*, diverging from all previous privacy audit work on understanding privacy deteriorating due to *composition*. Also motivated by our empirical findings, we further improve privacy upper bound specifically for a white-box application: the hyper-parameter tuning problem, pre-conditioned on the base algorithm, is DP-SGD. There is another notable work on private hyper-parameter tuning [15] by proposing a different algorithm with different assumptions than [42] and [30] (in [42] and [30], they only require the base algorithm to be DP; in [15], they must partition the training dataset to be disjoint). Since the starting point of this work is [42] and [30], we will only focus on the same line of [42] and [30].

III. Current Private Selection Protocol

A. Current Algorithm

We start with the state-of-the-art algorithm for private selection [30], [42], shown in Algorithm 2. This generic method applies when the base algorithm is already differentially private. When each $\mathcal{M}_i \in \Omega$ is a DP-SGD instance, the task becomes private hyperparameter tuning. If the base algorithm $\mathcal{M}$ is $(\varepsilon, 0)$-DP and ξ is geometric, then Algorithm 2 is $(3\varepsilon, 0)$-DP [30]. [42] improves this for pure DP by using a Truncated Negative Binomial (TNB) distribution for ξ under specific parameters (see Appendix B). The improvement uses RDP-based analysis.

Algorithm 2 Private Selection Protocol $\mathcal{H}$ [42], [30]

Input: Dataset X; algorithms Ω; distribution ξ; score function g

1: Draw a sample: $k \leftarrow \xi$
2: $Y \leftarrow$ **Null**, $S \leftarrow -\infty$
3: **for** $i = 1, 2, \cdots, k$ **do**
4: Uniformly randomly fetch one element $\mathcal{M}_i$ from Ω
5: $y_i \leftarrow \mathcal{M}_i(X)$ ▷ Run $\mathcal{M}_i$ on dataset X
6: **If** $g(y_i) > S$: $Y \leftarrow y_i$, $S \leftarrow g(y_i)$ ▷ Selecting the "best"
7: **end for**

Output: Y

B. Our General Tightness Proof

We show the current privacy upper bound due to [42] is tight in a general sense.

Example 1 (Our Construction for Pure DP)**.** *Let $\mathcal{M}$ have a finite output space $\mathcal{Y} = \{\mathrm{A}, \mathrm{B}, \mathrm{C}\}$. $\mathcal{M}$ only cares about the number of data samples in its input. If the number is even, its output follows the distribution shown as the left-hand side of Equation* (5)*; otherwise, its output distribution is the right-hand side.*

$$\Pr_{\mathcal{M}} \begin{cases} \Pr_{\mathrm{A}} = 1 - b\mathrm{e}^{\varepsilon} - db \\ \Pr_{\mathrm{B}} = b\mathrm{e}^{\varepsilon} \\ \Pr_{\mathrm{C}} = db \end{cases} \quad \Pr_{\mathcal{M}'} \begin{cases} \Pr_{\mathrm{A}'} = 1 - b - db\mathrm{e}^{\varepsilon} \\ \Pr_{\mathrm{B}'} = b \\ \Pr_{\mathrm{C}'} = db\mathrm{e}^{\varepsilon} \end{cases} \tag{5}$$

where $\Pr_{\mathrm{A}}$ denotes the probability of event A occurs conditioned on even (similarly we also have $\Pr_{\mathrm{A}'}$ with respect to odd). With $b = 10^{-3}, d = 10^2, \varepsilon = 1$, we can see $\mathcal{M}$ is clearly $(1, 0)$-DP for any pair of adjacent (w.r.t. addition/removal) dataset.

Let each element $\mathcal{M}_i$ fetched from Ω in line 4 of Algorithm 2 has the same output distribution as Equation (5)*. Also let a score function g give $g(\mathrm{C}) > g(\mathrm{B}) > g(\mathrm{A})$. Let ξ be the TNB distribution with parameter $\eta = 1, \nu = 10^{-3}$ (geometric distribution). The probability for each event that Algorithm 2 outputs is computed by the following.*

Claim 1. *Let y be some event in $\mathcal{Y}$, the probability of y occurs as the output of the tuning process $\mathcal{H}$ (Algorithm 2) is*

$$\Pr(y) = \sum_{k \sim \xi} \Pr(k) \left(\Pr(E_{\leq y})^k - \Pr(E_{<y})^k \right), \tag{6}$$

where $E_{\leq y} = \{x : g(x) \leq g(Y)\}$ and $E_{<y} = \{x : g(x) < g(Y)\}$. See proof in Appendix D.

Let $\Pr_{\mathcal{H}}, \Pr_{\mathcal{H}'}$ denote the probabilities for each event conditioned on $\mathcal{H}$ operates on adjacent dataset pair. For $\Pr_{\mathcal{H}}$ we have

$$\Pr_{\mathcal{H}} \begin{cases} \Pr_{\mathrm{A}|\mathcal{H}} = \sum_{k \sim \xi} \Pr(k) \Pr_{\mathrm{A}}^k \\ \Pr_{\mathrm{B}|\mathcal{H}} = \sum_{k \sim \xi} \Pr(k) ((\Pr_{\mathrm{A}} + \Pr_{\mathrm{B}})^k - \Pr_{\mathrm{A}}^k) \\ \Pr_{\mathrm{C}|\mathcal{H}} = \sum_{k \sim \xi} \Pr(k) (1 - (\Pr_{\mathrm{A}} + \Pr_{\mathrm{B}})^k) \end{cases}$$

where $\Pr_{\mathrm{A}|\mathcal{H}}$ denotes the probability of event A occurs as the output of $\mathcal{H}$ conditioned on the input dataset contains even number of data points. $\Pr_{\mathcal{H}'}$ can be computed similarly. Numerically, this gives the probabilities shown below

$$\Pr_{\mathcal{H}} \begin{cases} \Pr_{\mathrm{A}|\mathcal{H}} = 8.66 \times 10^{-3} \\ \Pr_{\mathrm{B}|\mathcal{H}} = 2.60 \times 10^{-4} \\ \Pr_{\mathrm{C}|\mathcal{H}} = 9.91 \times 10^{-1} \end{cases} \quad \Pr_{\mathcal{H}'} \begin{cases} \Pr_{\mathrm{A}|\mathcal{H}'} = 2.66 \times 10^{-3} \\ \Pr_{\mathrm{B}|\mathcal{H}'} = 1.34 \times 10^{-5} \\ \Pr_{\mathrm{C}|\mathcal{H}'} = 9.97 \times 10^{-1} \end{cases} \tag{7}$$

and it can be checked to satisfy $(2.96, 0)$-DP. The theoretical bound claims Algorithm 2 is $(3, 0)$-DP, i.e., it is tight up to a negligible gap.

Our example shows non-asymptotic tightness—an exact bound up to negligible error. This is more convincing than the example in [42, Appendix D.3], which relies on assumptions and first-order approximations. For approximate DP ($\delta > 0$), tightness also holds and is shown trivially in Appendix B.

This raises a new question: Does this worst-case tightness still apply when tuning hyper-parameters using multiple DP-SGD runs? We explore this in the following sections, focusing

on the case where each $\mathcal{M}_i \in \Omega$ is a DP-SGD instance with the same privacy guarantee. To begin, we conduct privacy audit experiments to examine how tight the previous bounds are in practice for DP-SGD in the next section.

Notation	Meaning
$\mathcal{G}$	The distinguishing game, Algorithm 1
$\mathcal{P}$	A general protocol to be audited in $\mathcal{G}$
$\mathcal{H}$	The private tuning protocol, Algorithm 2
$\mathcal{M}$	The base algorithm (DP-SGD) of $\mathcal{H}$
$\mathbb{F}, \mathbb{M}, \mathbb{C}, \mathbb{S}$	Datasets used, shown in Section IV-C
N	Number of iterations inside $\mathcal{M}$
C	Clipping threshold in Equation (1)
w_i	Model at i-th iteration in Equation (1)
ℓ	The loss function in Equation (1)
ξ	Running number distribution of $\mathcal{H}$
g	Score function evaluating $\mathcal{M}$'s output
$\mathbf{z}$	Differing data point, constructed by adversary
$p_i^{\mathbf{z}}$	$\mathbf{z}$'s gradient at i-th iteration in Equation (1)
p_i	Private gradient in Equation (1)
$\mathbf{Z}_D$	Hypothetical $\mathbf{z}$ leading to *Dirac* gradient
λ_a, λ_b	Two proxies constructed by the adversary
σ	Noise s.t.d. for R_i in Equation (1)
ε_B	Base algorithm $\mathcal{M}$'s privacy budget
ε_L	Lower bound for $\mathcal{H}$ by audit
ε_U	Generic upper bound for $\mathcal{H}$, by [42]

TABLE I: Notations used in our empirical study.

IV. Empirical Investigation

In this section, we aim to find how much privacy is leaked due to the tuning procedure $\mathcal{H}$ when the base algorithm is specifically the DP-SGD protocol. Notations used are summarised in Table I.

A. High-level Procedure

Simulate $\mathcal{G}$. We instantiate Algorithm 1 for our experiments, shown in Figure 1. Each execution of $\mathcal{P}$ in $\mathcal{G}$ is an execution of our tuning protocol $\mathcal{H}(\hat{X}, \Omega, \xi, g)$. Ω contains many base algorithms (DP-SGD instances with different hyper-parameter setups) satisfying the same privacy parameter. ξ is the TNB distribution [42] shown in Appendix B. g is the score function.
Conclude the lower bound. Our *null* and *alternative* hypothesis are

$$\mathbf{H_0} : X \text{ was used}, \quad \mathbf{H_1} : X' \text{ was used}. \tag{8}$$

After many simulations of $\mathcal{G}$ where each one gives an assertion for the above hypothesis testing problem, the FP and FN are computed by the Clopper-Pearson method [10] with a 95% confidence. We then leverage methods proposed in [37] to compute the empirical privacy lower bound $\varepsilon_L^{(X,X',\mathcal{R})}$. We provide the detailed procedure for deriving $\varepsilon_L^{(X,X',\mathcal{R})}$ in Appendix C. We omit the notation $(X, X', \mathcal{R})$ under clear context.

B. Audit Scenario Formulation

This section is to elucidate the four "arrows" originating from the adversary shown in Figure 1.

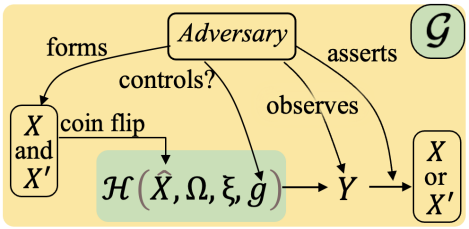


Fig. 1: Diagram of the distinguishing game $\mathcal{G}$.

Forming X, X'. W.o.l.g., we assume $X' = X \cup \{\mathbf{z}\}$. Note that the adversary can set X to be any available datasets. $\mathbf{z}$, known as "canaries" [37], is instantiated as follows.

- *Weaker version*. The adversary can select $\mathbf{z}$ to be any real-world data, and to have higher distinguishing performance, $\mathbf{z}$ is set to be sampled from a distribution different from those in X.
- *Stonger version*. The adversary can directly control the gradient of $\mathbf{z}$, *a.k.a.*, gradient canary. Specifically, it is assumed that adversary generates $\mathbf{z} = \mathbf{Z}_D$ such that its gradient is a *Dirac* vector $\nabla_w \ell(w; \mathbf{Z}_D) = [C, 0, 0, \cdots, 0]^T$ [37], i.e., only the first coordinate is C.

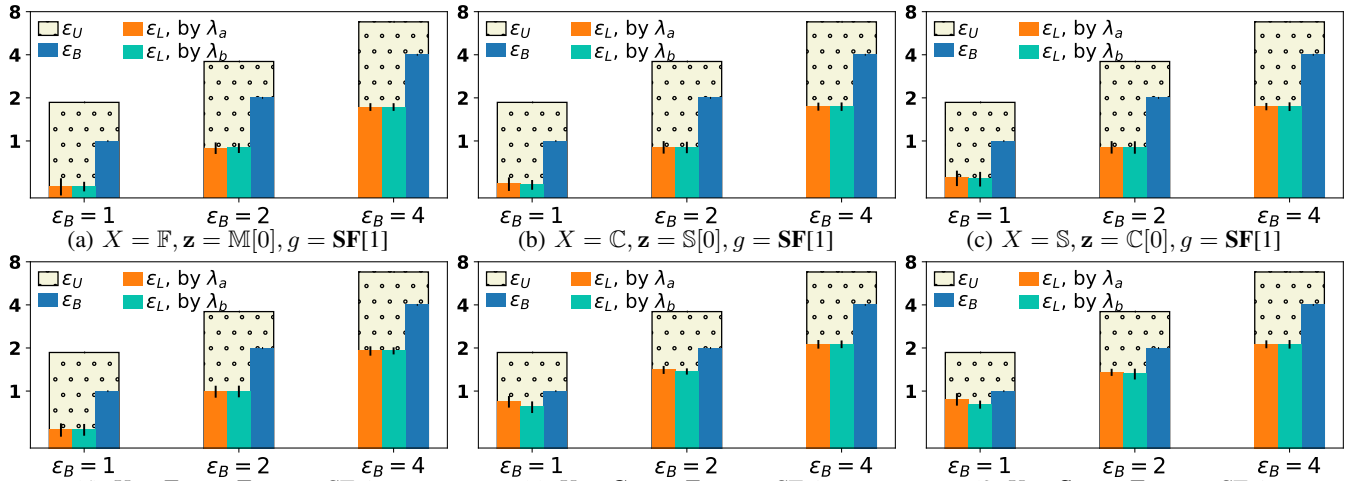
Score function g. The best model is selected if it has the *highest* score. *This new factor distinguishes auditing $\mathcal{H}$ from all previous auditing tasks*. Formalizing this factor and making corresponding assertions are our key contributions. We formalize two types of adversaries that are only possible.

- *Weaker version*. *g is not* manipulated, e.g., g is a normal routine to evaluate the model's accuracy/loss on an untampered validation dataset.
- *Stronger version*. The adversary can arbitrarily control g, e.g., g can be a routine to evaluate the model's performance on some malicious dataset. In practice, we believe the score function is some pre-defined function (e.g., the validation accuracy) and is not able to be manipulated. We enforce this setup is to explore the worst-case privacy leakage.

Adversary's observation Y. Under the assumption of DP-SGD protocol, the whole training trajectory $\{p_i\}_{i=1}^N$ is released. Equivalently, all the checkpoints $\{w_i\}_{i=1}^N$ of the neural network are trivially derivable as each checkpoint is just post-processing of the private gradient. Hence, we can denote the observation as $Y = \{p_1, p_2, \cdots p_N, w_1, \cdots, w_N\}$. This information corresponds to line 3 of Algorithm 1 or the output of $\mathcal{H}$. Note that including $w_i, i \in \{1, 2, \cdots, N\}$ in Y may be redundant; however, it is for notation convenience as we will later refer to the w_i information contained in Y.
Adversary's assertion. Adversary's assertion is exactly the action shown in line 4 in Algorithm 1. This requires the adversary to transform observations Y into binary guesses. The general procedure is as follows.

The adversary forms a real-number proxy based on observations and compares it to some threshold to make assertions. Proxies are described as follows:

A base proxy λ_a will be formed following previous work [39], [37] as follows. Compute $\mathbf{z}$'s gradient at each iteration



(a) $X = \mathbb{F}, \mathbf{z} = \mathbb{M}[0], g = \mathbf{SF}[1]$ (b) $X = \mathbb{C}, \mathbf{z} = \mathbb{S}[0], g = \mathbf{SF}[1]$ (c) $X = \mathbb{S}, \mathbf{z} = \mathbb{C}[0], g = \mathbf{SF}[1]$

(d) $X = \mathbb{F}, \mathbf{z} = \mathbf{Z}_D, g = \mathbf{SF}[1]$ (e) $X = \mathbb{C}, \mathbf{z} = \mathbf{Z}_D, g = \mathbf{SF}[1]$ (f) $X = \mathbb{S}, \mathbf{z} = \mathbf{Z}_D, g = \mathbf{SF}[1]$

Fig. 2: ***NTNV*** setup. Rows differ in differing data $\mathbf{z}$; columns differ in training datasets X. The vertical axis shows the values for ε_U and audited ε_L based on different proxies. Notations are explained by Equation (11).

before model update:

$$\lambda_a = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{C^2}\langle p_i^{\mathbf{z}}, p_i\rangle, \tag{9}$$

where $\langle a, b\rangle$ is the inner product. By design, other data examples are independent of $\mathbf{z}$. Hence, we expect $p_i^{\mathbf{z}}$ is likely to be (approximately) orthogonal to other data's gradient. Moreover, the adversary has access to the score function. Therefore, it seems reasonable to leverage such additional information. To this end, we will also form another proxy λ_b based on the score function as follows.

$$\lambda_b = \lambda_a - g(Y) \tag{10}$$

λ_b is our *newly* formed proxy and can be seen as the enhanced version of λ_a in auditing hyper-parameter tuning because it tries to include additional information from the score function g.

Summary. We form the following scenarios with increasing levels of threat.

- *Normal training and normal validation* (***NTNV***). The training dataset is a natural, normal dataset, and the training model's validation is also normal, i.e., score function g *is not* manipulated.
- *Normal training and controlled validation* (***NTCV***). The training dataset is the same as that of ***NTNV***; however, the validation for the trained model is controlled, i.e., score function g *is* manipulated.
- *Empty training and controlled validation* (***ETCV***). The training dataset is empty (malicious), g is the same as that of ***NTCV***.

C. Evaluation Methods

Given the complexity of this subject, here we describe how to evaluate our experimental result. The used dataset and our code link are provided in Appendix C. For notation convenience, we use abbreviations for the used datasets: $\mathbb{F}$ stands for the FASHION dataset, $\mathbb{M}$ for MNIST, $\mathbb{C}$ for CIFAR10 and $\mathbb{S}$ for SVHN. We use "[]" to fetch the information from some data container. For instance, we use $v[0]$ to denote fetching the first coordinate of v if v is a vector. We also abuse the notation and use $Y[w_N]$ to denote fetching the parameter w_N from output/observation Y.

Results indexing. Our main audit results are presented in figures, and we index them in the following form:

$$X = \mathbb{F}, \mathbf{z} = \mathbb{M}[I], g = \mathbf{SF}[1], \tag{11}$$

which means that such a result corresponds to 1) setting X to be the FASHION dataset; 2) setting the differing data $\mathbf{z}$ to be the I-th data sample from MNIST dataset; 3) setting the score function g to be the first candidate shown in Table II. Note that $X' = X \cup \{\mathbf{z}\}$ and we always shuffle the dataset initially.

Not manipulated	1: $g(Y) = -\sum_i \ell(Y[w_N]; \mathbb{V}[i])$ $\mathbb{V}$ is original validation dataset
Manipulated	2: $g(Y) = -\ell(Y[w_N]; \mathbf{z})$ 3: $g(Y) = (Y[w_0] - Y[w_N])[0]$

TABLE II: Score functions are indexed by $\mathbf{SF}[a], a = 1, 2, 3$.

Score function design consideration. As it will become clearer in later sections, the rationale behind manipulating the score function to be $\mathbf{SF}[2]$ is to expect the training to memorize (having low loss) the different data $\mathbf{z}$, and the best model is selected based on this metric. Manipulating the score function to be $\mathbf{SF}[3]$ builds on the fact that for $\mathbf{Z}_D$, it suffices only to investigate the first coordinate of the model to recover any trace of $\mathbf{z} = \mathbf{Z}_D$.

Evaluation method. Our main focus is to compare the following bounds; hence, understanding their intuitive interpretations is beneficial.

- ε_L is the amount of information leakage the adversary can extract based on the execution of $\mathcal{H}$.

- ε_B is the maximal information leakage due to a single run of the base algorithm, as guaranteed by theoretical analysis [35], [2].
- ε_U is the maximal information leakage due to execution of $\mathcal{H}$, as guaranteed by theoretical analysis [30], [42].

These bounds are all based on fixed σ values. Specifically, after σ is fixed for the base algorithm $\mathcal{M}$, we 1) compute ε_B by previous privacy analysis for DP-SGD such as TensorFlow privacy [1]; 2) compute ε_U for $\mathcal{H}$ by current generic bound for hyper-parameter tuning [42]; 3) apply privacy audit to $\mathcal{H}$, obtaining ε_L as shown in Section IV-A. We know that $\varepsilon_B \leq \varepsilon_U$ is always true, and it is interesting to make the following comparison.

- ε_L V.S. ε_U. This is the main focus. The question to be answered in this comparison is: does hyper-parameter tuning $\mathcal{H}$ practically leak sensitive information (ε_L) as predicted by the current generic bound [42] (ε_U)?
- ε_L V.S. ε_B. This is another interesting comparison. The question to be answered in this comparison is: How does running a DP-SGD many times and then returning the best (an execution of $\mathcal{H}$) practically leak information (ε_L) compared to a single run of DP-SGD (ε_B)?

D. Experiments When g Not Manipulated

***NTNV*.** This scenario corresponds to the most practical setup in our experiments. Experimental results are shown in Figure 2, notated according to Section IV-C.

Assertion intuition. The selection behaves normally, i.e., the best model is selected if it has the highest score (lowest loss) on the original validation dataset.

By design, higher value of λ_a or λ_b incentivizes the adversary to accept $\mathbf{H_1}$. The rationale behind these setups is to expect the abnormally differing data (if X' is used, or $\mathbf{z}$ was included in the training) to have a detrimental impact on the training so that the model has a higher loss (lower value of $g(Y)$), making it more distinguishable if X' is used ($\mathbf{z}$ was included in the training).

Results. Experimental results are presented in Figure 2, where we present audited ε_L results corresponding to proxy λ_a or λ_b. We also present the theoretical upper bound ε_U for comparison. An obvious phenomenon is that the audited $\varepsilon_L < \varepsilon_B < \varepsilon_U$ across all setups shown in the first row of Figure 2. The interesting phenomenon is that $\varepsilon_L < \varepsilon_B$ and the gaps between them are obvious.

In contrast, in Figure 2e and Figure 2f, when the base algorithm's privacy budget $\varepsilon_B = 1$, we see that ε_L is much closer to ε_B. This confirms that the differing data $\mathbf{z}$ that has *Dirac gradient* gains the adversary more distinguishing power than some natural data. Another phenomenon is that the audited ε_L under $\varepsilon_B = 1$ in Figure 2d is weaker than that in Figure 2e and Figure 2f, this suggests that auditing performance depends on X.

***Short summary*.** The adversary cannot even extract more sensitive information than the base algorithm's (a single run of DP-SGD) privacy budget allows, i.e., $\varepsilon_L < \varepsilon_B < \varepsilon_U$. This shows the adversary's power is heavily limited under the most practical setting.

E. Experiments When g Manipulated

NTCV*.** This scenario corresponds to some middle-level adversary's power. Experimental results are shown in Figure 3, notated according to Section IV-C. The score function *is* manipulated, different from that in ***NTNV.

Assertion intuition. The proxy λ_a is identical to that in Equation (9), however, $\lambda_b = \lambda_a + g(Y)$ is set in ***NTCV***, which is different from that in ***NTNV***. This is because g is manipulated. Under the same design considerations, a higher value of λ_a or λ_b incentivizes the adversary to accept $\mathbf{H_1}$.

Results. Experimental results are presented in Figure 3, organized similarly to Figure 2. We observe a phenomenon similar to ***NTNV*** that ε_L sees a big gap to ε_B shown in the first row of Figure 3. In contrast, for the results seen in the second row of Figure 3e and Figure 3f, when base algorithm's privacy budget $\varepsilon_B = 1$, we have $\varepsilon_L \approx \varepsilon_B$. Again, this confirms the *Dirac gradient* canary is more powerful.

Short summary. λ_a and λ_b have almost the same performance, similar to ***NTNV*** where g is not manipulated.

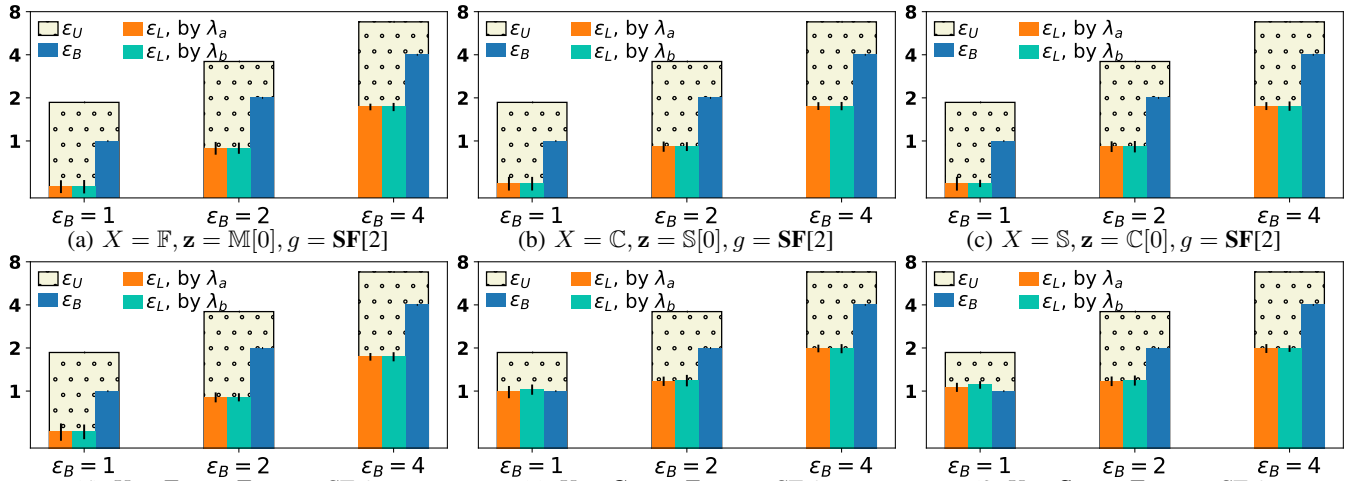
***ETCV*.** This scenario corresponds to the greatest adversary's power in our settings. The training dataset is set to be empty, and the score function is manipulated.

Assertion intuition. By design, the rationale behind the empty dataset setup is to eliminate the uncertainties due to normal training data's gradient so that audit performance is maximized, as the adversary only cares about the causal effect from $\mathbf{z}$ to the output [49]. λ_a, λ_b are set identically to ***NTCV***. Again, higher value of λ_a or λ_b incentivize the adversary to accept $\mathbf{H_1}$.

Results. Experimental results are shown in Figure 4, notated according to Section IV-C. In Figure 4a, we can see that $\varepsilon_L \geq \varepsilon_B$ under almost all setups; we also notice that ε_L still sees a gap to ε_U under some setups; however, ε_L gets much closer to ε_U compared with that in ***NTNV*** and ***NTCV***. The increased audit performance is due to $X = \emptyset$, which eliminates unwanted disturbances for the adversary. In Figure 4b, when $\mathbf{z} = \mathbf{z}_D$ is the *Dirac gradient* canary instead of some natural data, we observe $\varepsilon_L \geq \varepsilon_B$ under all setups.

The above results suggest that, operationally, hyper-parameter tuning does leak additional privacy beyond what's allowed to be disclosed by the base algorithm. This also means that tuning hyper-parameters while only accounting the privacy cost for a single run (i.e., naively taking $\varepsilon_U = \varepsilon_B$) is problematic in a rigorous manner. On the other hand, like the results in the previous two setups, we also observe that 1) λ_a and λ_b have almost the same performance, and 2) there is a big gap between ε_L and ε_U.

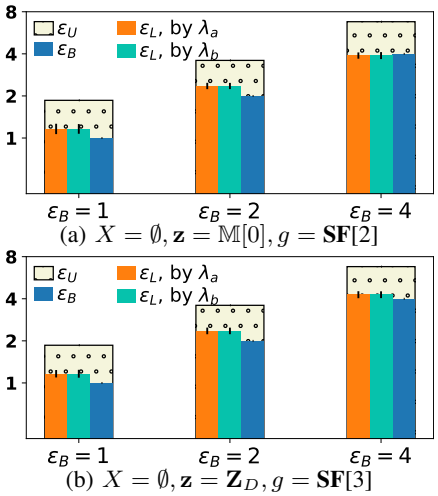
Short summary. Worst-case X setup does bring additional help to the adversary ($\varepsilon_L \approx \varepsilon_B$), but the privacy lower bound derived by audit still sees a gap to the privacy upper bound derived by previous work [42] ($\varepsilon_L < \varepsilon_U$).



(a) $X = \mathbb{F}, \mathbf{z} = \mathbb{M}[0], g = \mathbf{SF}[2]$ (b) $X = \mathbb{C}, \mathbf{z} = \mathbb{S}[0], g = \mathbf{SF}[2]$ (c) $X = \mathbb{S}, \mathbf{z} = \mathbb{C}[0], g = \mathbf{SF}[2]$

(d) $X = \mathbb{F}, \mathbf{z} = \mathbf{Z}_D, g = \mathbf{SF}[3]$ (e) $X = \mathbb{C}, \mathbf{z} = \mathbf{Z}_D, g = \mathbf{SF}[3]$ (f) $X = \mathbb{S}, \mathbf{z} = \mathbf{Z}_D, g = \mathbf{SF}[3]$

Fig. 3: ***NTCV*** setup. Sub-figure arrangement is identical to Figure 2.



(a) $X = \emptyset, \mathbf{z} = \mathbb{M}[0], g = \mathbf{SF}[2]$

(b) $X = \emptyset, \mathbf{z} = \mathbf{Z}_D, g = \mathbf{SF}[3]$

Fig. 4: ***ETCV*** setup. $\varepsilon_L \geq \varepsilon_B$ is almost true.

Proxy	ε_B	ε_U	ε_L						
			NTNV @ $X =$			***NTCV*** @ $X =$			***ETCV*** @ $X =$
			$\mathbb{F}$	$\mathbb{C}$	$\mathbb{S}$	$\mathbb{F}$	$\mathbb{C}$	$\mathbb{S}$	$\emptyset$
λ_a	1	1.86	0.54	0.84	0.88	0.52	0.99	1.06	1.17
λ_b			0.54	0.77	0.81	0.52	1.03	1.11	1.19
λ_a	2	3.59	0.99	1.41	1.35	0.91	1.17	1.17	2.36
λ_b			1.00	1.37	1.32	0.91	1.19	1.19	2.09
λ_a	4	6.79	1.91	2.13	2.13	1.74	1.99	1.99	4.28
λ_b			1.91	2.13	2.13	1.74	1.98	1.98	4.68

TABLE III: Results summary for privacy audit result ε_L (averaged over 5 runs) obtained under two different proxies when the canary $\mathbf{z} = \mathbf{z}_D$. We also highlight the base algorithm's upper bound ε_B and upper bound for hyper-parameter tuning using [42]'s method.

F. Phenomenon Highlight

New finding. Our ***NTNV*** and ***NTCV*** settings give a result that $\varepsilon_L < \varepsilon_U$, leading to the conclusion that weak audit leads to weak lower bound. Although our audit problem differs from the previous audit on DP-SGD's privacy, a weak audit leading to a weak lower bound is also observed previously. However, for ***ETCV*** setting (strong audit setting), **what we find is starkly different from the previous audit on DP-SGD**:

> *previous work [39] shows that maliciously chosen dataset X leads to $\varepsilon_L \approx \varepsilon_U$ (tight result) for auditing DP-SGD's privacy. In contrast, our audit on hyper-parameter tuning shows that ε_L still sees a noticeable gap to ε_U even when X is adversarily chosen.*

This new finding in auditing private hyper-parameter tuning motivates our theoretical study presented in the next section.

How does the score function (selection) leak privacy? This is a key question. Unlike prior auditing works [39], [23], [37], it has not been studied. Table III summarizes key audit results. First, λ_a and λ_b perform similarly across all settings. Second, comparing ***NTNV*** and ***NTCV*** shows that manipulating the score function offers no clear advantage to the adversary.

This may seem surprising. One might expect that tailoring the metric to pick the best candidate would make distinguishing easier. But our results show otherwise. This is because λ_a already reflects the effect of the score function—it is computed from the selected output. So λ_b adds no new information.

In fact, the score function output is post-processing of a DP algorithm, and thus remains differentially private. Whether or not $\mathbf{z}$ is included, the induced score distributions are close—just as DP guarantees. So no "magic" advantage exists, regardless of the score function, if it's independent of sensitive data.

Is it safe to tune hyper-parameter while only assuming a single run's privacy cost? Our weakest audit setup, which mimics practical use, suggests yes. Tuning adds little to no extra leakage beyond the base algorithm. This supports the common—but non-rigorous—practice of ignoring tuning overhead before private selection methods were formalized [30], [42].

However, under our strongest audit, tuning does leak more privacy. This confirms the importance of formally accounting for the tuning step, justifying both the previous work and our

improved bound in Section V.

Can we get a higher lower bound on any other datasets than what we get in *ETCV* setting? The *ETCV* setup already achieves dataset-independent lower bounds. Since the adversary can inject arbitrary gradient canaries, any bound achievable on real data is also achievable in this setting.

V. Improved Theoretical Results

In the previous empirical study, a conspicuous gap exists between ε_U derived by [42] and ε_L, this makes it interesting to investigate the reason behind it. Our study in the remaining sections shows that such a gap exists for two reasons.

1) Current generic privacy upper bound is not tight for DP-SGD;
2) Adversary's power is not strong enough because it is hard for the adversary to instantiate the worst-case score function g.

Regarding 1), we provide improved privacy results and elucidate on the special property of DP-SGD leading to the improvement; Our analysis is generalizable beyond DP-SGD, i.e., as will be shown, our analysis works for any base algorithm that can be expressed within the f-DP framework. For 2), we present meaningful findings about the score function.

Problem modelling. Informally, the privacy problem for our private tuning algorithm $\mathcal{H}$ (Algorithm 2) can be compactly described as the following optimization formulation.

$$\begin{aligned} &\textbf{\textit{minimize:}} && \varepsilon_{\mathcal{H}} \\ &\textbf{\textit{subject to:}} && \mathcal{H} \text{ satisfies } (\varepsilon_{\mathcal{H}}, \delta_{\mathcal{H}})\text{-DP given } \delta_{\mathcal{H}}; \\ & && \text{base algorithm's privacy is } \boldsymbol{\Lambda} \end{aligned} \tag{12}$$

It is self-evident that the tightness of $\varepsilon_{\mathcal{H}}$ depends on how tight $\boldsymbol{\Lambda}$ is. The critical part is how we represent the privacy guarantee $\boldsymbol{\Lambda}$. Under our optimization formulation, previous work describes $\boldsymbol{\Lambda}$ as follows: 1) Liu et.al [30] represents the base algorithm by (ε, δ)-DP; 2) Papernot et al. [42] does that by (α, γ)-RDP, obtaining improved results over [30]. Can we do better? As will be shown below, the answer is yes if we represent the base algorithm's privacy by f-DP.

A. Preliminaries: f-DP

f-DP [17], a privacy formulation with a finer resolution, reflects the nature of private mechanisms by a *function* [61] rather than a single pair of parameters. Our improved results are derived based on the f-DP framework. We introduce the necessary definitions and technical preliminaries in the following.

Definition 3 (Trade-off function [17])**.** *For a hypothesis testing problem over two distributions P, P', define the trade-off function as:*

$$T_{P,P'}(\mathrm{FP}) = \inf_{\mathcal{R}}\{\mathrm{FN}_{\mathcal{R}} : \mathrm{FP}_{\mathcal{R}} \leq \mathrm{FP}\}$$

where decision rule $\mathcal{R}$ takes input a sample from P or P' and decides which distribution produced that sample. The infimum is taken over all decision rule $\mathcal{R}$.

The trade-off function governs the best one can achieve when distinguishing P from P', i.e., by the optimal/smallest type II error (FN) at fixed type I error (FP). The optimal FN is achieved via the likelihood ratio test, which is also known as the fundamental *Neyman–Pearson lemma* [41] (please refer to Appendix A). We denote

$$g \geq f \text{ if } g(x) \geq f(x), \forall x \in [0,1].$$

Definition 4 (f-DP [17])**.** *Let $f : [0,1] \to [0,1]$ be a trade-off function. A mechanism $\mathcal{M}$ satisfies f-DP if*

$$T_{\mathcal{M}(X),\mathcal{M}(X')} \geq f$$

for all adjacent dataset pairs X, X'

$\mathcal{M}$ being f-DP means that any possible error pair $(\mathrm{FP}, \mathrm{FN})$ resulting from distinguishing $\mathcal{M}(X)$ from $\mathcal{M}(X')$ is lower-bounded by the curve specified by f. To see why (ε, δ)-DP is loose. We must express (ε, δ)-DP with the language of f-DP. This is done via the following proposition.

Proposition 1 ((ε, δ)-DP equals to $f_{\varepsilon,\delta}$-DP [17], [53])**.** *$\mathcal{M}$ is (ε, δ)-DP if and only if it is $f_{\varepsilon,\delta}$-DP where the trade-off function $f_{\varepsilon,\delta}$ is*

$$f_{\varepsilon,\delta}(x) = \max\left(0, 1-\delta-\mathrm{e}^{\varepsilon}x, \mathrm{e}^{-\varepsilon}(1-\delta-x)\right)$$

f-DP implies (ε, δ)-DP and conversion from f-DP to (ε, δ)-DP is via Algorithm 3 (restatement of Proposition 6 of [17]).

In plain words, $f_{\varepsilon,\delta}$-DP (or (ε, δ)-DP) for some mechanism $\mathcal{M}$ is the two symmetric straight lines lower-bounding the true/faithful trade-off function of $\mathcal{M}$. This is drawn in Figure 5. For the Gaussian mechanism, which is probably the most basic private mechanism, using (ε, δ)-DP to characterize its privacy is not tight/faithful; in contrast, the following special family of trade-off functions is tight.

Definition 5 (μ-Gaussian DP (μ-GDP) [17])**.** *The trade-off function of distinguishing $\mathcal{N}(0,1)$ from $\mathcal{N}(\mu,1)$ is*

$$G_\mu(x) = T_{\mathcal{N}(0,1),\mathcal{N}(\mu,1)}(x) = \Phi(\Phi^{-1}(1-x) - \mu),$$

where Φ be the c.d.f. of standard normal distribution. A private mechanism $\mathcal{M}$ satisfies μ-GDP if it is G_μ-DP

The analytical expression of μ-GDP is due to the application of Neyman–Pearson lemma on distinguishing $\mathcal{N}(0,1)$ from $\mathcal{N}(\mu,1)$ [17]. $\mathcal{M}$ satisfying μ-GDP means that distinguishing $\mathcal{M}(X)$ from $\mathcal{M}(X')$ is at least as hard as distinguishing $\mathcal{N}(0,1)$ from $\mathcal{N}(\mu,1)$. Figure 5 explains why (ε, δ)-DP is loose: (ε, δ)-DP is *strictly more conservative* than μ-GDP when characterizing the privacy of Gaussian mechanism.

Algorithm 3 f-DP to (ε, δ)-DP [17]

Input: f, trade-off function; δ, privacy parameter
1: If $\delta < 1 - f(0)$, return ∞
2: Compute $\varepsilon = \inf\{a : f(x) \geq 1 - \delta - \mathrm{e}^a x, \forall x \in [0,1]\}$ via binary search
Output: $\max\{0, \varepsilon\}$

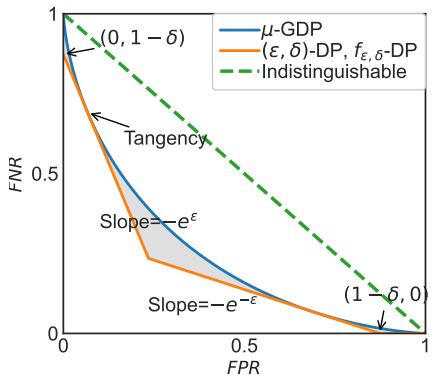


Fig. 5: For the Gaussian mechanism $\mathcal{M}(X) = q(X) + \mathcal{N}(0, \sigma^2\mathbb{I}^d)$ where the query function $q(X) \in \mathbb{R}^d$ has unit ℓ_2-sensitivity, it is exactly $1/\sigma$-GDP [17]. It is also some (ε, δ)-DP; however, μ-GDP characterization saves the shaded area in gray.

The following corollary gives a closed-form solution for optimal/lossless conversion from μ-GDP to $f_{\varepsilon,\delta}$-DP (or (ε, δ)-DP) in accordance with Algorithm 3.

Corollary 1 (Conversion from μ-GDP to (ε, δ)-DP formulation [17], [4])**.** *A mechanism is μ-GDP if and only if it is $f_{\varepsilon,\delta(\varepsilon)}$-DP (or $(\varepsilon, \delta(\varepsilon))$-DP) $\forall \varepsilon \geq 0$ where*

$$\delta(\varepsilon) = \Phi(-\frac{\varepsilon}{\mu} + \frac{\mu}{2}) - \mathrm{e}^{\varepsilon}\Phi(-\frac{\varepsilon}{\mu} - \frac{\mu}{2}) \tag{13}$$

Remark 1. *The purpose of introducing all previous technical preliminaries (especially Figure 5) is not only to necessarily introduce f-DP itself but also to understand why we can obtain improvements under f-DP framework (see Example 2).*

DP-SGD is asymptotic μ-GDP. μ-GDP is pivotal because it asymptotically characterizes the privacy of any DP-SGD instance having many compositions of Gaussian mechanisms [17]:

Corollary 2 (GDP approximation [17] for DP-SGD)**.** *DP-SGD is asymptotically μ-GDP with*

$$\mu = \sqrt{2}\tau\sqrt{N} \cdot \sqrt{\mathrm{e}^{\sigma^{-2}} \cdot \Phi\left(1.5\sigma^{-1}\right) + 3\Phi\left(-0.5\sigma^{-1}\right) - 2}$$

where $\sigma = \sigma'/C$ and σ' is s.t.d. of the Gaussian noise; C is the clipping threshold; τ and N is the sampling ratio and number of total iteration of DP-SGD.

B. Our Contribution: Improved Results

A critical fact about Corollary 2 is that computing the exact trade-off function is #P-hard [17] (even harder than NP problems), which makes it necessary to resort to approximations if one aims at tighter results within the f-DP framework. Specifically, the error (*pointwise error* between the asymptotical GDP trade-off function and the true trade-off function) decays at a rate of $1/\sqrt{N}$ for DP-SGD, shown by [17]. Therefore, using Corollary 2 requires N to be large enough. Such a condition holds for probably most DP-SGD applications, especially for training large models (e.g., $N > 10^4$ in [2] and $N > 10^5$ in [27], [51]).

Based on all of the above preparations, we are ready to approach our privacy problem by filling in the missing part of Equation (12) based on Corollary 2:

$$\begin{aligned} &\textbf{\textit{minimize:}} && \varepsilon_{\mathcal{H}} \\ &\textbf{\textit{subject to:}} && \mathcal{H} \text{ satisfies } (\varepsilon_{\mathcal{H}}, \delta_{\mathcal{H}})\text{-DP given } \delta_{\mathcal{H}}; \\ & && \text{base algorithm's privacy is } \mu\text{-GDP} \end{aligned} \tag{14}$$

In the following, we first revisit the central question of how selection (the score function) leaks privacy.

In Section IV-F, we showed that the adversary gains no advantage even if the score function g is maliciously manipulated. A natural question arises: is there any score function that brings more advantage to the adversary?

One-to-one mapping g is the worst-case necessarily. The score function g is a function that maps the output of the base algorithm to a real number, If some g happens to map two distinct inputs to the same score (hence, a randomized tie-breaking will be enforced), how does such g affect the privacy of $\mathcal{H}$ compared to one-to-one mapping score functions?

Intuitively, such g will only make $\mathcal{H}$ more private as new uncertainty is injected. We can gain more intuition by considering the extreme case: if g only outputs a constant, then $\mathcal{H}$ is just as private as the base algorithm. Our theorem in the following formalizes such intuition.

Theorem 2 (Necessary worst-case g, proof in Appendix E)**.** *Let distribution P be over some finite alphabets Γ, and define a distribution $F_{k,g}$ as follows.*

First, make $k > 0$ independent samples $\{x_1, x_2, \cdots, x_k\}$ from P; second, output x_i such that the score $g(x_i)$ computed by a score function $g : \Gamma \to \mathbb{R}$ is the maximum over these samples. Similarly, we define another distribution P' over the same alphabets Γ and derive a distribution $F'_{k,q}$ as the counterpart to $F_{k,g}$.

*For any score function $\hat{g}$, which is **not** a one-to-one mapping (hence a randomized tie-breaking is needed), there always exists a one-to-one mapping g^* satisfying*

$$\mathcal{D}_\alpha(F_{k,\hat{g}} || F'_{k,\hat{g}}) \leq \mathcal{D}_\alpha(F_{k,g^*} || F'_{k,g^*}). \tag{15}$$

Moreover, similar inequality also holds when k follows a general distribution ξ.

The above result is derived under RDP (Definition 2) and it tells us crucial facts: *A score function that induces a strict total order for elements in Γ tends to be less private*. Thus, a one-to-one mapping is necessary to be the worst case for the score function g.

Note that, in previous work [30], [42], the score function g is assumed to be one-to-one mapping *by default for simplicity*. We show that such treatment is valid due to privacy considerations; to our knowledge, the above theorem is the first rigorous proof validating such an assumption.

Theorem 2 also holds when Γ is infinite because Rényi divergence can be approximated arbitrarily well by finite

partition [50, Theorem 10]. With g's necessary condition determined, we can introduce our improved privacy results.

Notation. Let $y, y' \in \mathcal{Y}$ be the output of the base algorithm (DP-SGD, a single run) corresponding to adjacent input dataset X, X', respectively. Let P, P' be the induced score distribution after the score function g takes input y, y', respectively. With some abuse of notation, we use $P(x), F(x)$ to denote the p.d.f. and c.d.f. for distribution P (similarly, we have $P'(x), F'(x)$ w.r.t. X'). Based on the assumption that g is a one-to-one mapping, the selection is essentially among samples from P (or P' if X' is the input).

Let Q be the distribution of the score of the model outputted by $\mathcal{H}$. Let us for now consider the distribution ξ in $\mathcal{H}$ is a point mass on some $k > 0$, i.e., $\Pr(k) = 1$. Then, the p.d.f. $Q(x)$ is

$$Q(x) = kP(x)(F(x))^{k-1} \tag{16}$$

as well-studied in *order statistics* [13], i.e., it is the distribution of the maximal sample among k independent draws.

When distribution ξ is some general distribution, define the function

$$\omega_\xi(x) = \sum_{k\sim\xi} k \cdot \Pr_\xi(k) \cdot x^{k-1} \tag{17}$$

and then Q is a mixture distribution, i.e.,

$$Q(x) = \sum_{k\sim\xi} \Pr_\xi(k) \cdot kP(x)(F(x))^{k-1} = P(x)\omega_\xi(F(x)). \tag{18}$$

Distribution Q''s p.d.f. corresponding to X' being the input is computed similarly. Now, we are ready to present our improved privacy upper bound.

Theorem 3 (General form, proof in Appendix F)**.** *Suppose the base algorithm is f-DP, then $\mathcal{H}$ is $(\varepsilon_\mathcal{H}, \delta_\mathcal{H})$-DP where*

$$\varepsilon_\mathcal{H} = \varepsilon + \max_{a\in[0,1]} \log \frac{\omega_\xi(1-a)}{\omega_\xi(b)}, \tag{19}$$

where $b = f(a)$ and ε is computed by Algorithm 3 whose two input arguments are the trade-off function f and $\delta = \delta_\mathcal{H}/\omega_\xi(1)$ (ω_ξ is defined in Equation (17)*).*

We present our f-DP accountant for private selection in Algorithm 4 according to Theorem 3.

Algorithm 4 f-DP Accountant for $\mathcal{H}$

Input: trade-off function f s.t. the base algorithm is f-DP, ξ distribution of $\mathcal{H}$, $\delta_\mathcal{H}$

1: $\delta \leftarrow \delta_\mathcal{H}/\omega_\xi(1)$ ▷ ω_ξ is from Equation (17)
2: $\varepsilon \leftarrow$ input f and δ to Algorithm 3
3: $\varepsilon_\mathcal{H} \leftarrow \varepsilon + \max_{a\in[0,1]} \log\left(\omega_\xi(1-a)/\omega_\xi(f(a))\right)$

Output: $\varepsilon_\mathcal{H}$

Given that the base algorithm is some μ-GDP, we immediately arrive at the improved result for hyper-parameter tuning by plugging in its specific trade-off function.

Corollary 3 (Improved result for DP-SGD)**.** *If the base algorithm if μ-GDP (or G_μ-DP), then $\mathcal{H}$ is $(\varepsilon_\mathcal{H}, \delta_\mathcal{H})$-DP where*

$$\varepsilon_\mathcal{H} = \varepsilon + \max_{a\in[0,1]} \log \frac{\omega_\xi(1-a)}{\omega_\xi(G_\mu(a))} \tag{20}$$

with $G_\mu(a)$ is in Definition 5 and $\delta_\mathcal{H}/\omega_\xi(1) = \Phi(-\frac{\varepsilon}{\mu} + \frac{\mu}{2}) - \mathrm{e}^\varepsilon\Phi(-\frac{\varepsilon}{\mu} - \frac{\mu}{2})$ determines ε.

Intuitive explanation of our results. The intuition is that the selection (choosing the best of many independent runs) results in a different output distribution than when running the base algorithm only once. And it will deteriorate the final privacy bound, this is shown in Equation 19: there is an increase (deteriorating) of the ε parameter compared to the base algorithm's parameter. How the results deteriorate depends on ξ.

Modeling the base algorithm with f-DP instead of (ε, δ)-DP leads to tighter bounds. By the post-processing property [17], the score output retains the same f-DP as the base algorithm. In Equation (19), a represents the false negative (FN), and $b = f(a)$ is the optimal false positive (FP) at that FN.

If the base algorithm satisfies μ-GDP, then $b = G_\mu(a)$. But if modeled using (ε, δ)-DP, we only get $b = f_{\varepsilon,\delta}(a)$. Figure 5 shows that $G_\mu(a) \geq f_{\varepsilon,\delta}(a)$. Since ω_ξ is increasing, this gives a tighter (smaller) $\varepsilon_\mathcal{H}$ when using GDP. The following example illustrates the gain from using f-DP.

Example 2. *Suppose the base algorithm (DP-SGD) satisfies 1-GDP and ξ is the TNB distribution with parameter $\eta = 1, \nu = 10^{-2}$ (in this case, ξ is geometric distribution, and we recover the case studied by Liu et al. [30]). Hence, it allows us to make meaningful comparisons.*

For $\delta = 10^{-5}$, the base algorithm is also $(4.36, 10^{-5})$-DP or $f_{4.36,10^{-5}}$-DP. If $b = G_1(a)$ in Equation (19)*, which is how we represent the base algorithm's privacy, we have $\max_{a\in[0,1]} \log \frac{\omega_\xi(1-a)}{\omega_\xi(G_1(a))} = 3.3$; however, if $b = f_{4.36,10^{-5}}(a)$, which equals to how the base algorithm is modeled by Liu et al. [30], $\max_{a\in[0,1]} \log \frac{\omega_\xi(1-a)}{\omega_\xi(f_{4.36,10^{-5}}(a))} = 16.5 > 3.3$ is only what we can derive. Thus, a huge improvement is obtained, and this is due to the saved shaded area in gray shown in Figure 5.*

C. Significance Statement

Our result is generalizable and tighter. We observe that 1) [30] only supports geometric ξ, and 2) [42] only supports truncated negative binomial and Poisson ξ. It is unclear how to handle arbitrary ξ, and prior results require manual, case-specific analysis. In contrast, our result works for any ξ in protocol $\mathcal{H}$. Computing ω_ξ is always numerically stable, as $\omega_\xi(x)$ is bounded on $[0, 1]$.

As shown in Section VI-B, our bound improves over prior work. For example, if ξ always outputs 1 (i.e., run the base algorithm once), Equation (19) gives $\varepsilon_\mathcal{H} = \varepsilon$ and $\delta_\mathcal{H} = \delta$, matching the base guarantee. This shows our result is tight for general ξ. In contrast, RDP-based analysis is loose due to lossy conversion to (ε, δ)-DP [61].

Extension beyond DP-SGD. Our above example shows that representing the privacy of the base algorithm with finer resolution (from (ε, δ)-DP to f-DP) leads to improvements in the privacy upper bound. Similar conclusions also hold when switching from RDP [42] to f-DP as RDP is also observed to be lossy within the f-DP framework [61], i.e., RDP shares

the same weakness as that of the (ε, δ)-DP. We select DP-SGD as our base algorithm because of its popularity in the literature, but our result is not limited to DP-SGD. In fact, any private base algorithm analyzed by (ε, δ)-DP or RDP can be represented by f-DP with finer resolution. And switching to f-DP and using our privacy accountant can also bring improvements. The reason is depicted in Figure 5: using f-DP avoids the unnecessary region shaded in gray.

VI. Further Evaluation

A. Stronger Audit via Reduction

Motivation for final audit trial. In the presence of our improved privacy upper bound, we immediately want to assess its tightness by privacy audit for general ξ such that $\Pr_\xi(1) < 1$. This requires we derive reasonably strong lower bounds to be informative. We should avoid ad hoc audit setups for real-world training tasks (Section IV). We need to form our audit with theoretical-justified power.

This section is to serve such a purpose. A part of the design considerations relies on our Theorem 2 in the last section.

1) Base algorithm reduction. Our threat model will be based on the assumption made by DP, i.e., the adversary knows the membership of all data used to update the model in each iteration except for the differring data $\mathbf{z}$ (the *strong adversary assumption* [12], [29]).

This means the adversary can always subtract the gradient of other data from p_i in each iteration. Hence, any adjacent dataset pair X, X' is equivalent to $X = \emptyset, X' = \mathbf{z}$ from the adversary's view. This allows us to make two reduction steps for the base algorithm (DP-SGD).

❶ *First reduction* (from Equation (21) to Equation (23)). Let σ noise s.t.d. shown in Equation (1). Now assume that the sampling ratio $\tau = 1$, i.e., full-batch gradient descent. Given $X = \emptyset, X' = \mathbf{z}$, then, at each iteration, for the adversary, the private gradient p_i is as follows.

$$\begin{aligned} p_i|X &= R_i \sim \mathcal{N}(0, C^2\sigma^2\mathbb{I}^d) \\ p_i|X' &= (\nabla_{\mathbf{z}} + R_i) \sim \mathcal{N}(\nabla_{\mathbf{z}}, C^2\sigma^2\mathbb{I}^d), \end{aligned} \tag{21}$$

where $p_i|X$ denotes the random variable conditioned on X was chosen and $\nabla_{\mathbf{z}} = \nabla_w \ell(w_{i-1}, z)$ with $\|\nabla_{\mathbf{z}}\|_2 = C$ (assume maximal ℓ_2-norm is reached). The adversary *can always construct a rotational matrix* $U_{\mathbf{z}} \in \mathbb{R}^{d\times d}$ such that p_i can be reduced as follows.

$$U_{\mathbf{z}} p_i|X \sim \mathcal{N}(0, C^2\sigma^2\mathbb{I}^d), \quad U_{\mathbf{z}} p_i|X' \sim \mathcal{N}(U_{\mathbf{z}}\nabla_{\mathbf{z}}, C^2\sigma^2\mathbb{I}^d) \tag{22}$$

where $U_{\mathbf{z}}\nabla_{\mathbf{z}} = [C, 0, 0, \cdots]^T$. This is because Gaussian noise with $\sigma^2\mathbb{I}^d$ covariance is rotational invariant, i.e.,

$$\mathbf{Cov}(U_{\mathbf{z}} R_i) = U_{\mathbf{z}}\mathbf{Cov}(R_i)U_{\mathbf{z}}^T = C^2\sigma^2\mathbb{I}^d = \mathbf{Cov}(R_i)$$

where $\mathbf{Cov}(R_i)$ is the covariance matrix of Gaussian random vector R_i. After the rotation, for the adversary, only the first coordinate carries useful information about $\mathbf{z}$.

Because a noise vector $R_i \sim \mathcal{N}(0, C^2\sigma^2\mathbb{I}^d)$ and its rotated version $U_{\mathbf{z}} R_i \sim \mathcal{N}(0, C^2\sigma^2\mathbb{I}^d)$ possessing $\sigma^2\mathbb{I}^d$ covariance matrix are all coordinate-wise independent. To serve the distinguishing purpose: $\mathbf{z}$ was/was not used, it suffices to characterize the private gradient p_i by $\bar{p}_i$ as a univariate random variable (the first coordinate) for the distinguishing purpose as follows.

$$\bar{p}_i|X \sim \mathcal{N}(0, C^2\sigma^2), \quad \bar{p}_i|X' \sim \mathcal{N}(C, C^2\sigma^2). \tag{23}$$

❷ *Second reduction* (from Equation (23) to Equation (25)). Base on previous reduction, the DP-SGD's output $\bar{y} = \{\bar{p}_1, \bar{p}_2, \cdots, \bar{p}_N\}$ is essentially an observation of N i.i.d. samples from $\mathcal{N}(a, C^2\sigma^2)$ where a is either 0 or C. Recall the adversary's goal is to distinguish X or X' was used; this is equivalent to determining $a = 0$ or $a = C$.

As both distributions in Equation (23) are Gaussian, we can use the *sufficient statistics* for estimating a [21], [11], which is the mean: $\bar{y} = \frac{1}{N}\sum_{i=1}^N \bar{p}_i$. Sufficient statistics do not lose any information for estimating a. Finally, we can reduce the privacy of the base algorithm to an equivalent game for the adversary as

$$\bar{y}|X \sim \mathcal{N}(0, C^2\sigma^2/N), \quad \bar{y}|X' \sim \mathcal{N}(C, C^2\sigma^2/N). \tag{24}$$

For simplicity, applying a simple invertible/lossless re-scaling gives us equivalent characterization:

$$\bar{y}|X \sim \mathcal{N}(0, 1), \quad \bar{y}|X' \sim \mathcal{N}(\sqrt{N}/\sigma, 1). \tag{25}$$

There is a slight difference in the reduction when $\tau < 1$. Instead of arriving at Equation (23), we arrive at

$$\bar{p}_i|X \sim \mathcal{N}(0, C^2\sigma^2), \quad \bar{p}_i|X' \sim \mathcal{N}(Cb_i, C^2\sigma^2), \tag{26}$$

where $b_i, \forall i \in \{1, 2, \cdots, N\}$ is independent Bernoulli random variables with probability τ. By doing the same transformation as from Equation (23) to Equation (25), we arrive at

$$\bar{y}|X \sim \mathcal{N}(0, 1), \quad \bar{y}|X' \sim \mathcal{N}(\frac{\sum_{i=1}^N b_i}{N}\sqrt{N}/\sigma, 1). \tag{27}$$

Equation (27) also covers Equation (25) when $\tau = 1$.

2) Instantiate the score function. Based on our above reduction, to serve the distinguishing purpose, a model obtained by DP-SGD can be "treated" as a real number sampled from univariate Gaussian or its shifted counterpart corresponding to X or X' was used. The order induced by the score function g is now over $\mathbb{R}$. To have stronger audit results in our idealized attack, we need to instantiate the worst-case score function, and our Theorem 2 tells us one-to-one mapping score function g is the worst case necessarily.

However, Theorem 2 remains silent on the specific analytical form of g in the worst case. There can be infinitely many one-to-one mapping functions $\mathbb{R} \to \mathbb{R}$; for implementation purposes, we now fix a score function $g(x) = x$, i.e., we take g is strictly increasing and note that all strictly increasing functions induces the same order over $\mathbb{R}$ regardless of its analytical form. Finally, we present the distinguishing game of our reduced case in Figure 6, which will be simulated many times, allowing high-confident conclusions of the lower bound following Section II-B.

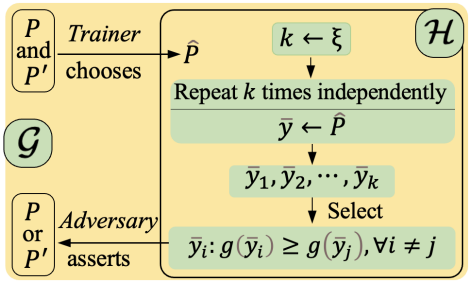


Fig. 6: The distinguishing game for $\mathcal{H}$ by reduction. P, P' are two univariate Gaussians shown in Equation (27), corresponding to adjacent dataset pair X, X'. The adversary then makes binary assertions by comparing the best sample to some threshold. The game $\mathcal{G}$ is simulated 10^7 times.

B. Privacy Results Comparison

Privacy results. We present the lower bound ε_L, our improved privacy results $\varepsilon^O_{\mathcal{H}}$ and previous privacy results $\varepsilon^P_{\mathcal{H}}$ [42] in Table IV. We can see that the audit is rather powerful: $\varepsilon_L > \varepsilon_B$ with a noticeable margin, once again confirming the very action of selecting the best does leak additional privacy beyond the base algorithm's privacy budget. On the other hand, our privacy result $\varepsilon^O_{\mathcal{H}}$ is better than the previous result $\varepsilon^P_{\mathcal{H}}$ under all setups.

Implication. The improvement we obtain is much more significant than the $\epsilon^O_{\mathcal{H}}$ value shows cosmetically. For instance, the results due to our improved upper bound $\epsilon^O_{\mathcal{H}}$ under $(\eta = 1, \nu = 10^{-3})$ are even consistently smaller than $\varepsilon^P_{\mathcal{H}}$ of previous upper bound under $(\eta = 1, \nu = 10^{-2})$. Thus, our improved analysis can allow trying significantly more hyper-parameter candidates while even consuming less privacy budget. For private hyper-parameter tuning applications, this translates to improved utility of the trained model for free.

To show the downstream gain using our method, we present the privacy result comparison in Table V. As can be seen from Table V, under the same privacy constraint, we allow more running (in expectation) hyper-parameter candidates, and the final achievable testing accuracy is consistently better as expected.

C. Lessons Learned and Open Problems

As shown in Section V-C, our improved result is indeed tight for general ξ in terms of how much $(\varepsilon_{\mathcal{H}}, \delta_{\mathcal{H}})$-DP $\mathcal{H}$ satisfies. However, we still see a noticeable gap between our result $\varepsilon^O_{\mathcal{H}}$ and the lower bound ε_L derived by our idealized attack. why does this happen?

Our answer is that g plays a critical role from the adversary's point of view, and such a factor distinguishes attacking private hyper-parameter tuning from all previous privacy attack problems. We have shown that one-to-one mapping is necessary for g being the worst case, but there are infinitely many g over an infinite output domain and are up to $|\Gamma|!$ possible choices if output domain Γ is finite. We find that some of the score functions leak more privacy than others. For example, when

ε_B	τ	$\varepsilon_L \mid \varepsilon^O_{\mathcal{H}} \mid \varepsilon^P_{\mathcal{H}}$			
		$\eta = 0, \nu = 10^{-2}$ $\mathbb{E}_\xi = 21$	$\eta = 1, \nu = 10^{-2}$ $\mathbb{E}_\xi = 100$	$\eta = 1, \nu = 10^{-3}$ $\mathbb{E}_\xi = 1000$	$\eta = 2, \nu = 10^{-3}$ $\mathbb{E}_\xi = 2000$
1	1	1.17 | 1.55 | 1.86	1.19 | 2.06 | 2.65	1.27 | 2.54 | 3.09	1.29 | 3.18 | 3.99
	0.5	1.09 | 1.53 | 1.87	1.30 | 2.04 | 2.64	1.37 | 2.51 | 3.15	1.42 | 3.15 | 4.05
	0.1	1.00 | 1.49 | 1.86	1.19 | 1.98 | 2.56	1.49 | 2.43 | 3.24	1.46 | 3.05 | 4.09
2	1	2.17 | 2.92 | 3.61	2.21 | 3.84 | 5.06	2.53 | 4.69 | 5.89	2.57 | 5.85 | 7.57
	0.5	2.05 | 2.89 | 3.57	2.34 | 3.80 | 4.93	2.59 | 4.64 | 5.88	2.63 | 5.79 | 7.49
	0.1	2.16 | 2.90 | 3.58	2.40 | 3.82 | 4.83	2.87 | 4.67 | 6.03	2.86 | 5.82 | 7.53
4	1	3.93 | 5.70 | 6.80	4.32 | 7.40 | 9.30	4.51 | 8.95 | 10.83	4.70 | 11.07 | 13.77
	0.5	3.84 | 5.64 | 6.71	4.23 | 7.32 | 9.03	4.89 | 8.86 | 10.74	4.91 | 10.96 | 13.51
	0.1	3.76 | 5.48 | 6.58	4.17 | 7.12 | 8.64	5.02 | 8.62 | 10.61	5.03 | 10.67 | 13.08

TABLE IV: Comparison of privacy bounds for $\mathcal{H}$ (Algorithm 2). All values are in $(\varepsilon, \delta = 10^{-5})$-DP form. ε_L is the empirical lower bound obtained by simulating the distinguishing game in Figure 6. $\varepsilon^O_{\mathcal{H}}$ is our improved analytical upper bound. $\varepsilon^P_{\mathcal{H}}$ is the upper bound from prior work [42].Each row corresponds to a different sampling ratio τ, with total iterations fixed at $N = 10^3$. The parameters η and ν define the TNB distribution used to generate the number of runs in $\mathcal{H}$ (details in Appendix B), and $\mathbb{E}_\xi$ is the expected number of runs under this distribution.

ε_B	$\varepsilon^O_{\mathcal{H}}$	Previous → Ours			
		MNIST	FMNIST	CIFAR10	SVHN
1	1.83	0.921 → 0.934	0.768 → 0.793	0.412 → 0.448	0.636 → 0.661
2	3.43	0.942 → 0.956	0.779 → 0.802	0.467 → 0.486	0.706 → 0.745
4	6.69	0.951 → 0.958	0.791 → 0.817	0.504 → 0.531	0.762 → 0.786

TABLE V: Testing accuracy comparison under differentially private hyper-parameter tuning. The TNB setup for our method is $(\eta = 0, \nu = 10^{-3})$ $(\mathbb{E}_\xi \approx 144)$. To achieve roughly the same privacy result using previous method [42], the setup should be $(\eta = 0, \nu = 10^{-2})$ $(\mathbb{E}_\xi \approx 21)$. $\varepsilon^O_{\mathcal{H}}$ is our improved result for private hyper-parameter tuning. For all experiments, $\delta = 10^{-5}$.

$\eta = 1, \nu = 10^{-2}$ (TNB ξ recovers geometric distribution), if we (arbitrarily) set the score function as

$$g(x) = \begin{cases} x, & \text{for } x \in [-\infty, 0) \bigcup (1, \infty] \\ 1 - x, & \text{for } x \in [0, 1] \end{cases} \tag{28}$$

which is clearly a one-to-one mapping, we only derive $\varepsilon_L = 2.01$ (average of 10 runs) at $\varepsilon_B = 2, \tau = 1$, which is smaller/weaker than the value 2.21 shown in Table IV (where $g(x) = x$).

Choosing some g arbitrarily and performing the attack will likely end up with sub-optimal attacks (smaller/weaker lower bounds). This is probably the reason why we still see a gap between $\varepsilon^O_{\mathcal{H}}$ and ε_L even in our idealized attack. Reasoning on such issues is non-trivial, and it poses the following questions worthy of investigation:

1) which g should the adversary choose to elicit more privacy leakage? 2) does the worst-case g depend on specific ξ? 3) How to quantify the exact trade-off between privacy leakage and ξ which governs the utility? Answering the above questions requires non-trivial efforts, which we hold as meaningful future directions.

VII. Conclusion

We study how selection leaks privacy. Initially, we give examples showing that the current generic bound for private selection is indeed tight in general. Still, it is not tight for a white-box setting, i.e., the hyper-parameter tuning problem. Substantiating this assertion, we first audit the privacy of hyper-parameter tuning under various settings; the derived empirical privacy lower bound under the strongest adversary still sees a noticeable gap from the generic upper bound.

We then provide an in-depth study of deriving better privacy upper bound by modeling the base algorithm's privacy with finer resolution (f-DP). The improvement is due to the distinct properties of the base algorithm (DP-SGD). Our result allows trying many more hyper-parameter candidates while consuming less private budget. Our analysis also generalizes, contrasting with previous work, which remains unknown how to adapt to general parameter setups.

VIII. Ethics Considerations

This paper is on refining the privacy bound for differentially private protocols, not on privacy attacks. The privacy audit experiments conducted are to conclude a privacy lower bound, not to launch some real-world privacy attacks. All analyses and experiments are conducted using publicly available datasets to minimize privacy risks. Our study aims to strengthen differential privacy protections in hyper-parameter tuning by improving the analysis rather than exploiting any weaknesses.

Acknowledgment

Di Wang and Zihang Xiang are supported in part by the funding BAS/1/1689-01-01, URF/1/4663-01-01, REI/1/5232-01-01, REI/1/5332-01-01, and URF/1/5508-01-01 from KAUST, and funding from KAUST - Center of Excellence for Generative AI, under award number 5940. Tianhao Wang is supported by NSF CNS-2319988 and a CCI research grant.

References

[1] TensorFlow Privacy. https://github.com/tensorflow/privacy.

[2] 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*, pages 308–318, 2016.

[3] Meenatchi Sundaram Muthu Selva Annamalai, Georgi Ganev, and Emiliano De Cristofaro. " what do you want from theory alone?" experimenting with tight auditing of differentially private synthetic data generation. *arXiv preprint arXiv:2405.10994*, 2024.

[4] Borja Balle and Yu-Xiang Wang. Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In *International Conference on Machine Learning*, pages 394–403. PMLR, 2018.

[5] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *2014 IEEE 55th annual symposium on foundations of computer science*, pages 464–473. IEEE, 2014.

[6] Benjamin Bichsel, Timon Gehr, Dana Drachsler-Cohen, Petar Tsankov, and Martin Vechev. Dp-finder: Finding differential privacy violations by sampling and optimization. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 508–524, 2018.

[7] Benjamin Bichsel, Samuel Steffen, Ilija Bogunovic, and Martin Vechev. Dp-sniper: Black-box discovery of differential privacy violations using classifiers. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 391–409. IEEE, 2021.

[8] Karan Chadha, Matthew Jagielski, Nicolas Papernot, Christopher Choquette-Choo, and Milad Nasr. Auditing private prediction. *arXiv preprint arXiv:2402.09403*, 2024.

[9] Kamalika Chaudhuri and Staal A Vinterbo. A stability-based validation procedure for differentially private machine learning. *Advances in Neural Information Processing Systems*, 26, 2013.

[10] Charles J Clopper and Egon S Pearson. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, 26(4):404–413, 1934.

[11] Thomas M. Cover and Joy A. Thomas. *Elements of information theory (2. ed.)*. Wiley, 2006.

[12] Paul Cuff and Lanqing Yu. Differential privacy as a mutual information constraint. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 43–54, 2016.

[13] Herbert A David and Haikady N Nagaraja. *Order statistics*. John Wiley & Sons, 2004.

[14] 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.

[15] Youlong Ding and Xueyang Wu. Revisiting hyperparameter tuning with differential privacy. *arXiv preprint arXiv:2211.01852*, 2022.

[16] Zeyu Ding, Yuxin Wang, Guanhong Wang, Danfeng Zhang, and Daniel Kifer. Detecting violations of differential privacy. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 475–489, 2018.

[17] Jinshuo Dong, Aaron Roth, and Weijie Su. Gaussian differential privacy. *Journal of the Royal Statistical Society*, 2021.

[18] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*, pages 265–284. Springer, 2006.

[19] Cynthia Dwork, Moni Naor, Omer Reingold, Guy N Rothblum, and Salil Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 381–390, 2009.

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

[21] Ronald A Fisher. On the mathematical foundations of theoretical statistics. *Philosophical transactions of the Royal Society of London. Series A, containing papers of a mathematical or physical character*, 222(594-604):309–368, 1922.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016.

[23] Matthew Jagielski, Jonathan Ullman, and Alina Oprea. Auditing differentially private machine learning: How private is private sgd? *Advances in Neural Information Processing Systems*, 33:22205–22216, 2020.

[24] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. The composition theorem for differential privacy. In *International conference on machine learning*, pages 1376–1385. PMLR, 2015.

[25] Antti Koskela and Tejas Kulkarni. Practical differentially private hyperparameter tuning with subsampling. *arXiv preprint arXiv:2301.11989*, 2023.

[26] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[27] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[28] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[29] Ninghui Li, Wahbeh Qardaji, Dong Su, Yi Wu, and Weining Yang. Membership privacy: A unifying framework for privacy definitions. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 889–900, 2013.

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

[31] Fred Lu, Joseph Munoz, Maya Fuchs, Tyler LeBlond, Elliott Zaresky-Williams, Edward Raff, Francis Ferraro, and Brian Testa. A general

framework for auditing differentially private machine learning. *Advances in Neural Information Processing Systems*, 35:4165–4176, 2022.
[32] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 94–103. IEEE, 2007.
[33] Ilya Mironov. Rényi differential privacy. In *2017 IEEE 30th computer security foundations symposium (CSF)*, pages 263–275. IEEE, 2017.
[34] Ilya Mironov. Rényi differential privacy. In *30th IEEE Computer Security Foundations Symposium, CSF 2017, Santa Barbara, CA, USA, August 21-25, 2017*, pages 263–275. IEEE Computer Society, 2017.
[35] Ilya Mironov, Kunal Talwar, and Li Zhang. Rényi differential privacy of the sampled gaussian mechanism. *CoRR*, abs/1908.10530, 2019.
[36] 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, pages 7806–7813, 2022.
[37] Milad Nasr, Jamie Hayes, Thomas Steinke, Borja Balle, Florian Tramèr, Matthew Jagielski, Nicholas Carlini, and Andreas Terzis. Tight auditing of differentially private machine learning. In Joseph A. Calandrino and Carmela Troncoso, editors, *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*, pages 1631–1648. USENIX Association, 2023.
[38] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 739–753. IEEE, 2019.
[39] Milad Nasr, Shuang Songi, Abhradeep Thakurta, Nicolas Papernot, and Nicholas Carlin. Adversary instantiation: Lower bounds for differentially private machine learning. In *2021 IEEE Symposium on security and privacy (SP)*, pages 866–882. IEEE, 2021.
[40] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.
[41] Jerzy Neyman and Egon Sharpe Pearson. Ix. on the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 231(694-706):289–337, 1933.
[42] Nicolas Papernot and Thomas Steinke. Hyperparameter tuning with renyi differential privacy. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.
[43] Hanpu Shen, Cheng-Long Wang, Zihang Xiang, Yiming Ying, and Di Wang. Differentially private non-convex learning for multi-layer neural networks. *arXiv preprint arXiv:2310.08425*, 2023.
[44] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017.
[45] Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. Stochastic gradient descent with differentially private updates. In *2013 IEEE global conference on signal and information processing*, pages 245–248. IEEE, 2013.
[46] Thomas Steinke. Composition of differential privacy & privacy amplification by subsampling. *arXiv preprint arXiv:2210.00597*, 2022.
[47] Thomas Steinke, Milad Nasr, and Matthew Jagielski. Privacy auditing with one (1) training run. *arXiv preprint arXiv:2305.08846*, 2023.
[48] Qiaoyue Tang and Mathias Lécuyer. Dp-adam: Correcting dp bias in adam's second moment estimation. *arXiv preprint arXiv:2304.11208*, 2023.
[49] Michael Carl Tschantz, Shayak Sen, and Anupam Datta. Sok: Differential privacy as a causal property. In *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020*, pages 354–371. IEEE, 2020.
[50] Tim Van Erven and Peter Harremos. Rényi divergence and kullback-leibler divergence. *IEEE Transactions on Information Theory*, 60(7):3797–3820, 2014.
[51] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
[52] Yuxin Wang, Zeyu Ding, Daniel Kifer, and Danfeng Zhang. Checkdp: An automated and integrated approach for proving differential privacy or finding precise counterexamples. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 919–938, 2020.
[53] Larry Wasserman and Shuheng Zhou. A statistical framework for differential privacy. *Journal of the American Statistical Association*, 105(489):375–389, 2010.
[54] Zihang Xiang, Tianhao Wang, Wanyu Lin, and Di Wang. Practical differentially private and byzantine-resilient federated learning. *Proceedings of the ACM on Management of Data*, 1(2):1–26, 2023.
[55] Zihang Xiang, Tianhao Wang, Wanyu Lin, and Di Wang. Practical differentially private and byzantine-resilient federated learning. *Proc. ACM Manag. Data*, 1(2):119:1–119:26, 2023.
[56] Zihang Xiang, Tianhao Wang, and Di Wang. Preserving node-level privacy in graph neural networks. In *IEEE Symposium on Security and Privacy, SP 2024, San Francisco, CA, USA, May 19-23, 2024*, pages 4714–4732. IEEE, 2024.
[57] Zihang Xiang, Tianhao Wang, and Di Wang. Privacy audit as bits transmission:(im) possibilities for audit by one run. In *USENIX Security*, 2025.
[58] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017.
[59] Hanshen Xiao, Zihang Xiang, Di Wang, and Srinivas Devadas. A theory to instruct differentially-private learning via clipping bias reduction. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 2170–2189. IEEE Computer Society, 2023.
[60] Santiago Zanella-Béguelin, Lukas Wutschitz, Shruti Tople, Ahmed Salem, Victor Rühle, Andrew Paverd, Mohammad Naseri, Boris Köpf, and Daniel Jones. Bayesian estimation of differential privacy. In *International Conference on Machine Learning*, pages 40624–40636. PMLR, 2023.
[61] Yuqing Zhu, Jinshuo Dong, and Yu-Xiang Wang. Optimal accounting of differential privacy via characteristic function. In *International Conference on Artificial Intelligence and Statistics*, pages 4782–4817. PMLR, 2022.

APPENDIX

A. Neyman–Pearson Lemma

Theorem 4 (Neyman–Pearson lemma [41])**.** *Let P and Q be probability distributions on Ω with densities p and q, respectively. Define $L(x) = \frac{p(x)}{q(x)}$. For hypothesis testing problem*

$$\mathbf{H_0} : P, \quad \mathbf{H_1} : Q$$

For a constant $c > 0$, suppose that the likelihood ratio test which rejects $\mathbf{H_0}$ when $L(x) \leq c$ has $\mathrm{FP} = a$ and $\mathrm{FN} = b$, then for any other test of $\mathbf{H_0}$ with $\mathrm{FP} \leq a$, the achievable FN is at least b.

Neyman–Pearson lemma says that the most powerful test (optimal FN) at fixed FP is the likelihood ratio test. Applying Neyman–Pearson lemma to distinguishing $\mathcal{N}(0,1)$ from $\mathcal{N}(\mu,1)$ gives us Definition 5 [17].

B. Privacy Results by Papernot et al.

Truncated negative binomial (TNB) distribution. For $\nu \in (0,1)$ and $\eta \in (-1,\infty)$, the distribution $\xi_{\eta,\nu}$ on $\{1,2,3,\cdots\}$ is as follows. When $\eta \neq 0$, then

$$\forall k \in \mathbb{N}, \quad \Pr[K=k] = \frac{(1-\nu)^k}{\nu^{-\eta}-1} \cdot \prod_{\ell=0}^{k-1} \left(\frac{\ell+\eta}{\ell+1}\right),$$

when $\eta = 0$, then

$$\forall k \in \mathbb{N}, \quad \Pr[K=k] = \frac{(1-\nu)^k}{k \cdot \log(1/\nu)}$$

This particular distribution is obtained by differentiating the probability generating function of some desired form [42].

The main relevant privacy results in [42] are provided in the following. Note that they are all in RDP form.

Theorem 5 (RDP for TNB distribution [42])**.** *Let k in Algorithm 1 follows TNB distribution $\xi_{\eta,\nu}$. If the base algorithm satisfies (α,γ)-RDP and (α',γ')-RDP, Algorithm 1 satisfies $(\alpha,\hat{\gamma})$-RDP where*

$$\hat{\gamma} = \gamma + (1+\eta)\cdot\left(1-\frac{1}{\alpha'}\right)\gamma' + \frac{(1+\eta)\cdot\log(1/\nu)}{\alpha'} + \frac{\log \mathbb{E}_{\xi_{\eta,\nu}}}{\alpha-1}$$

Tight example for approximate DP. A Tight example for (ε,δ)-DP case can be obtained trivially based on Example 1. If an algorithm is $(\varepsilon,0)$-DP, it is also (ε,δ)-DP. Hence, the above tight example covers the (ε,δ)-DP case. Specifically, it can be checked that the example shown in Equation (5) is $(1,10^{-5})$-DP and Equation (7) is $(2.92,10^{-5})$-DP. Compared to the result predicted by [42], which is $(3.11,10^{-5})$-DP, i.e., it is tight up to a negligible gap.

C. Used Datasets and Experimental Details

Our implementation is provided at an Github link[1] (or an perminante link at "doi.org/10.5281/zenodo.17073774"). We use four image datasets in our experiments. FASHION [58], MNIST [28], CIFAR10 [26] and SVHN [40]. All of our experiments are conducted under privacy parameter $\delta = 10^{-5}$. The number of repeating/simulation times in an audit experiment is $2,000$. The error bar results from taking the min., max., and avg. for three trials. To efficiently audit the hyper-parameter tuning and reduce the simulation burden, we only fetch 5,000 data examples from the original training datasets, and we set the sampling rate to be 1, i.e., full gradient descent. We set the TNB distribution [42] with parameter $(\eta=0,\nu=10^{-2})$. We use the ResNet [22] as the neural network in our experiments. We use *Adam* as the default optimizer. The computational burden is significant: our audit experiment consumes >4000 GPU hours and is conducted over 20 GPUs in parallel.

Hyperparameter candidates setup. To run the audit experiments, we need to set the candidates inside Ω in Algorithm 2. We hold the clipping threshold C, learning rate $\mathbf{lr}$, and the number of total iterations N as the hyperparameters to be tuned. To form each candidate inside Ω, we randomly sample a value to determine C, $\mathbf{lr}$ and N. All candidates have the same privacy budget according to our problem formulation.

Detailed procedure of concluding the lower bound ε_L. The following procedure is adopted to conclude a lower bound ε_L.

1) **Generating** $(b_{\textbf{truth}}, b_{\textbf{guess}})$. Each pair corresponds to an execution of $\mathcal{G}$ (Algorithm 1). The adversary needs to make an assertion, i.e., output a $b_{\text{guess}} \in \{0,1\}$.

2) **Compute** ε_L. After getting many pairs of $(b_{\text{truth}}, b_{\text{guess}})$, the FP,FN can be summarised by Clopper-Pearson with a confidence c. Specifically, the FP rate and FN rate are modeled as the unknown success probabilities of two binomial distributions. Then ε_L can be computed by Equation (4) or by the methods used in [37].

[1]https://github.com/zihangxiang/PrivateHyperparameterTuning.git

3) **Optimization**. In practice, practitioners often try various assertion strategies on the same observed output by repeating procedures 1) and 2) to find the optimal ε_L.

D. Proof of Claim 1

Proof. when $k>0$ is some fixed integer, we know that the scores of all k runs are $\leq g(Y)$, which has the probability $\Pr(E_{\leq y})^k$. As y occurs, we have probability $\Pr(E_{\leq y})^k - \Pr(E_{<y})^k$ seeing y as the output of Algorithm 2. When k follows some general distribution ξ, the resultant distribution is a mixture, which is Claim 1. □

E. Proof of Theorem 2

Proof. W.o.l.g., we define the alphabets of distribution P and P' as $\{a,b,c,d,e,f\}$, with some abuse of notation, we denote

$$P(a)=p_a, P(b)=p_b, \cdots, P(f)=p_f$$
$$P'(a)=p'_a, P'(b)=p'_b, \cdots, P'(f)=p'_f$$

as their probabilities. Suppose we have a non-one-to-one mapping score evaluator $\hat{g}$ such that:

$$\hat{g}(a)=\hat{g}(c)=\hat{g}(e)<\hat{g}(b)=\hat{g}(d)<\hat{g}(f).$$

We now assume a uniformly random selection among Alphabets that share the same score. For clearer presentation, we denote $\Lambda_S^k = (\sum_{i\in S} p_i)^k$ and $\bar{\Lambda}_S^k = (\sum_{i\in S} p'_i)^k$. Then, the distribution of $F_{k,\hat{g}}$ will be

$$F_{k,\hat{g}}(a)=F_{k,\hat{g}}(c)=F_{k,\hat{g}}(e)=\frac{1}{3}\Lambda^k_{\{a,c,e\}}$$
$$F_{k,\hat{g}}(b)=F_{k,\hat{g}}(d)=\frac{1}{2}(\Lambda^k_{\{a,c,e,b,d\}}-\Lambda^k_{\{a,c,e\}})$$
$$F_{k,\hat{g}}(f)=1-\Lambda^k_{\{a,c,e,b,d\}}$$

This is because

$$F_{k,\hat{g}}(i\in\{a,c,e\})=\Lambda^k_{\{a,c,e\}}$$
$$F_{k,\hat{g}}(i\in\{b,d\})=\Lambda^k_{\{a,c,e,b,d\}}-\Lambda^k_{\{a,c,e\}}$$

and a uniformly random selection among $\{a,c,e\}$ means that the probability mass $F_{k,\hat{g}}(i\in\{a,c,e\})$ is distributed uniformly to each. Similarly, the distribution of $F'_{k,\hat{g}}$ corresponding to P' has the same form (just replace Λ_S^k by $\bar{\Lambda}_S^k$). We now construct a one-to-one mapping score function g^* as follows.

$$g^*(a)<g^*(c)<g^*(e)<g^*(b)<g^*(d)<g^*(f)$$

The key point here is to **enforce a strict total order for alphabets that have the same score**. Then, the distribution of F_{k,g^*} is

$$F_{k,g^*}(a)=\Lambda^k_{\{a\}}, F_{k,g^*}(c)=\Lambda^k_{\{a,c\}}-\Lambda^k_{\{a\}}$$
$$F_{k,g^*}(e)=\Lambda^k_{\{a,c,e\}}-\Lambda^k_{\{a,c\}}, F_{k,g^*}(b)=\Lambda^k_{\{a,c,e,b\}}-\Lambda^k_{\{a,c,e\}}$$
$$F_{k,g^*}(d)=\Lambda^k_{\{a,c,e,b,d\}}-\Lambda^k_{\{a,c,e,b\}}, F_{k,g^*}(f)=1-\Lambda^k_{\{a,c,e,b,d\}}$$

Similarly, the distribution of F'_{k,g^*} corresponding to P' has the same form (just replace Λ_S^k by $\bar{\Lambda}_S^k$). We now compute the RDP quantity $\mathcal{D}_\alpha(F_{k,\hat{g}}||F'_{k,\hat{g}})$ and $\mathcal{D}_\alpha(F_{k,g^*}||F'_{k,g^*})$. We aim to show that the RDP value under non-one-to-one mapping $\hat{g}$

is smaller than that under its one-to-one mapping counterpart. We group the sub-terms of RDP calculation. Let

$$\hat{T}_{\{a,c,e\}} = \sum_{i\in\{a,c,e\}} \left(\frac{F_{k,\hat{g}}(i)}{F'_{k,\hat{g}}(i)}\right)^{\alpha} F'_{k,\hat{g}}(i)$$
$$\hat{T}_{\{b,d\}} = \sum_{i\in\{b,d\}} \left(\frac{F_{k,\hat{g}}(i)}{F'_{k,\hat{g}}(i)}\right)^{\alpha} F'_{k,\hat{g}}(i)$$
$$\hat{T}_{\{f\}} = \left(\frac{F_{k,\hat{g}}(f)}{F'_{k,\hat{g}}(f)}\right)^{\alpha} F'_{k,\hat{g}}(f)$$

We compute the $T^*_{\{a,c,e\}}, T^*_{\{b,d\}}, T^*_{\{f\}}$ counterparts in the same fashion (just replace $\hat{g}$ by g^*). And we will compare $T_{\{a,c,e\}}$ and $T'_{\{a,c,e\}}$. By letting

$$\begin{aligned} x &= \Lambda^k_{\{a\}} & x' &= \bar{\Lambda}^k_{\{a\}} \\ y &= \Lambda^k_{\{a,c\}} - \Lambda^k_{\{a\}} & y' &= \bar{\Lambda}^k_{\{a,c\}} - \bar{\Lambda}^k_{\{a\}} \\ z &= \Lambda^k_{\{a,c,e\}} - \Lambda^k_{\{a,c\}} & z' &= \bar{\Lambda}^k_{\{a,c,e\}} - \bar{\Lambda}^k_{\{a,c\}} \end{aligned}$$

then, it is easy to see that

$$\begin{aligned} \hat{T}_{\{a,c,e\}} =& (\frac{\Lambda^k_{\{a,c,e\}}}{\bar{\Lambda}^k_{\{a,c,e\}}})^{\alpha} \bar{\Lambda}^k_{\{a,c,e\}} \\ =& (\frac{x+y+z}{x'+y'+z'})^{\alpha}(x'+y'+z') \\ \leq& (\frac{x}{x'})^{\alpha}x' + (\frac{y}{y'})^{\alpha}y' + (\frac{z}{z'})^{\alpha}z' \\ =& T^*_{a,c,e} \end{aligned} \tag{29}$$

holds by Jensen's inequality and the fact that function $h(x) = x^{\alpha}$ is convex for $\alpha > 1$, i.e.,

$$(\frac{x+y+z}{x'+y'+z'})^{\alpha} \leq \frac{(\frac{x}{x'})^{\alpha}x' + (\frac{y}{y'})^{\alpha}y' + (\frac{z}{z'})^{\alpha}z'}{x'+y'+z'}$$

For the same reason, it can also be easily checked that $\hat{T}_{\{b,d\}} \leq T^*_{\{b,d\}}$ and $\hat{T}_{\{f\}} \leq T^*_{\{f\}}$ also hold. Because

$$\mathcal{D}_{\alpha}(F_{k,\hat{g}}||F'_{k,\hat{g}}) = \frac{1}{\alpha-1}\ln(\hat{T}_{\{a,c,e\}} + \hat{T}_{\{b,d\}} + \hat{T}_{\{f\}})$$
$$\mathcal{D}_{\alpha}(F_{k,g^*}||F'_{k,g^*}) = \frac{1}{\alpha-1}\ln(T^*_{\{a,c,e\}} + T^*_{\{b,d\}} + T^*_{\{f\}}),$$

we have

$$\mathcal{D}_{\alpha}(F_{k,\hat{g}}||F'_{k,\hat{g}}) \leq \mathcal{D}_{\alpha}(F_{k,g^*}||F'_{k,g^*}).$$

Note the first equality of Equation (29) always holds no matter whether selection among alphabets sharing the same score is uniform or weighted; the alphabet and the order we choose is also arbitrary, which means that the result holds in general.

Remark. Following the same reasoning, when k is now a random variable instead of a fixed number, we also have the result, as shown in the above theorem. Because we can modify each probability term to be the probability of the mixture counterpart, and the proof follows trivially. Specifically, for each probability $p = f(k)$ shows up, modify it to be $p = \sum_{i=1}^{\infty} \Pr(i)f(i)$ where $\Pr(i), i = \{1, 2, \cdots, \infty\}$ is the p.m.f. of distribution ξ. □

F. Proof of Theorem 3

Proof. As we care about how much $(\varepsilon_{\mathcal{H}}, \delta_{\mathcal{H}})$-DP $\mathcal{H}$ satisfies given some $\delta_{\mathcal{H}}$, it is useful to introduce a technical lemma related to such form of DP.

Lemma 1 ([46] Propostion 7)**.** *Define the privacy loss random variable for a pair of adjacent dataset X, X' to a private mechanism $\mathcal{M}$ as $L_1 = \log\frac{\mathcal{M}(X)(o)}{\mathcal{M}(X')(o)}$ where $o \sim \mathcal{M}(X)$ $\mathcal{M}$ is (ε, δ)-DP or $f_{\varepsilon,\delta}$-DP if and only if*

$$\int_{\varepsilon}^{\infty} \mathrm{e}^{\varepsilon - z} \cdot \Pr_{o\sim\mathcal{M}(X)}[L_1 > z]\mathrm{d}z \leq \delta$$

holds for all adjacent X, X'.

Our goal is clear, i.e., we need to meet the following equation:

$$\int_{\varepsilon_{\mathcal{H}}}^{\infty} \mathrm{e}^{\varepsilon_{\mathcal{H}} - z} \cdot \Pr_{o\sim Q}[\log\frac{Q(o)}{Q'(o)} > z]\mathrm{d}z \leq \delta_{\mathcal{H}} \tag{30}$$

Then $\mathcal{H}$ would be $(\varepsilon_{\mathcal{H}}, \delta_{\mathcal{H}})$-DP.

Let the left-hand side of Equation (30) to be $t_{\varepsilon_{\mathcal{H}}}$ and note that $\frac{Q(o)}{Q'(o)} = \frac{P(o)\omega_{\xi}(F(o))}{P'(o)\omega_{\xi}(F'(o))}$, define event $E_z = \{o : \log\frac{P(o)\omega_{\xi}(F(o))}{P'(o)\omega_{\xi}(F'(o))} > z\}$ then

$$\begin{aligned} t_{\varepsilon_{\mathcal{H}}} =& \int_{\varepsilon_{\mathcal{H}}}^{\infty} \mathrm{e}^{\varepsilon_{\mathcal{H}} - z} \cdot \int\limits_{E_z} Q(o)\mathrm{d}o\mathrm{d}z \\ \leq& \omega_{\xi}(1) \int_{\varepsilon_{\mathcal{H}}}^{\infty} \mathrm{e}^{\varepsilon_{\mathcal{H}} - z} \cdot \int\limits_{E_z} P(o)\mathrm{d}o\mathrm{d}z \end{aligned} \tag{31}$$

The inequality is due to $\omega_{\xi} : [0,1] \to \mathbb{R}$ is increasing.

Let us investigate the hypothesis testing problem P V.S. P', i.e., deciding X or X' was used based on the score of a single run of the DP-SGD. The score is post-processing [17, Lemma 1]) of the trained model, so the $(\mathrm{FP}, \mathrm{FN})$ pair for distinguishing P from P' is governed by f.

For some real number $o \in \mathbb{R}$, define $A = \{u : u \leq o\}$, and a decision rule $\mathcal{R}$ that accepts P when the score falls into A. Then, $\mathrm{FP}_{\mathcal{R}} = 1 - F(o)$ and $\mathrm{FN}_{\mathcal{R}} = F'(o)$. And we must have $F'(o) \geq f(1 - F(o))$ as governed by the trade-off function. This leads to an upper bound (note that ω_{ξ} is increasing)

$$\frac{\omega_{\xi}(F(o))}{\omega_{\xi}(F'(o))} \leq \max_{a\in[0,1]} \frac{\omega_{\xi}(1-a)}{\omega_{\xi}(f(a))} = M \tag{32}$$

Now, let $\hat{E}_z = \{o : \log\frac{P(o)}{P'(o)}M > z\}$, it is easy to see that $E_z \subseteq \hat{E}_z$. Hence, we have

$$\begin{aligned} t_{\varepsilon_{\mathcal{H}}} \leq& \omega_{\xi}(1) \int_{\varepsilon_{\mathcal{H}}}^{\infty} \mathrm{e}^{\varepsilon_{\mathcal{H}} - z} \cdot \int\limits_{\hat{E}_z} P(o)\mathrm{d}o\mathrm{d}z \\ =& \omega_{\xi}(1) \int_{\varepsilon_{\mathcal{H}}}^{\infty} \mathrm{e}^{\varepsilon_{\mathcal{H}} - z} \cdot \Pr_{o\sim P}[\log\frac{P(o)}{P'(o)} > z - \log M]\mathrm{d}z \\ \leq& \omega_{\xi}(1) \int_{\varepsilon_{\mathcal{H}} - \log M}^{\infty} \mathrm{e}^{\varepsilon_{\mathcal{H}} - z} \cdot \Pr_{o\sim P}[\log\frac{P(o)}{P'(o)} > z - \log M]\mathrm{d}z \\ =& \omega_{\xi}(1) \int_{\varepsilon_{\mathcal{H}} - \log M}^{\infty} \mathrm{e}^{\varepsilon_{\mathcal{H}} - \log M - s} \cdot \Pr_{o\sim P}[\log\frac{P(o)}{P'(o)} > s]\mathrm{d}s \end{aligned}$$

Letting $s = z - \log M$, we have the last equality. Note that the score is differentially private, as it is post-processing of the base algorithm. Hence, we can compute a $(\varepsilon_{\mathcal{H}} - \log M, \delta)$-DP guarantee for the score. Applying Lemma 1, we have

$$t_{\varepsilon_{\mathcal{H}}} \leq \omega_{\xi}(1)\delta.$$

Setting $\delta_{\mathcal{H}} = \omega_{\xi}(1)\delta$, we derive δ. By inputting trade-off function f for the base algorithm and δ to Algorithm 3, we derive the value of $\epsilon_{\mathcal{H}} - \log M$, which give us Theorem 3. As we assume nothing on the adjacent dataset X, X', Theorem 3 holds for all X, X' pair.

□