To Shuffle or not to Shuffle: Auditing DP-SGD with Shuffling

Meenatchi Sundaram Muthu Selva Annamalai
University College London
meenatchi.annamalai.22@ucl.ac.uk

Borja Balle
Google Deepmind
bballe@google.com

Jamie Hayes
Google Deepmind
jamhay@google.com

Emiliano De Cristofaro
University of California, Riverside
emilianodc@cs.ucr.edu

***Abstract*—The Differentially Private Stochastic Gradient Descent (DP-SGD) algorithm supports the training of machine learning (ML) models with formal Differential Privacy (DP) guarantees. Traditionally, DP-SGD processes training data in batches using Poisson subsampling to select each batch at every iteration. More recently, shuffling has become a common alternative due to its better compatibility and lower computational overhead. However, computing tight theoretical DP guarantees under shuffling remains an open problem. As a result, models trained with shuffling are often evaluated as if Poisson subsampling were used, which might result in incorrect privacy guarantees.**

This raises a compelling research question: can we verify whether there are gaps between the theoretical DP guarantees reported by state-of-the-art models using shuffling and their actual leakage? To do so, we define novel DP-auditing procedures to analyze DP-SGD with shuffling and measure their ability to tightly estimate privacy leakage vis-à-vis batch sizes, privacy budgets, and threat models. Overall, we demonstrate that DP models trained using this approach have considerably overestimated their privacy guarantees (by up to 4 times). However, we also find that the gap between the theoretical Poisson DP guarantees and the actual privacy leakage from shuffling is not uniform across all parameter settings and threat models. Finally, we study two common variations of the shuffling procedure that result in even further privacy leakage (up to 10 times). Overall, our work highlights the risk of using shuffling instead of Poisson subsampling in the absence of rigorous analysis methods.

I. Introduction

To mitigate privacy risks [9, 49, 57], the Differentially Private Stochastic Gradient Descent (DP-SGD) [1] algorithm is increasingly being used to train models with Differential Privacy (DP) guarantees [24]. More precisely, DP provably bounds the leakage from a model so that no adversary can confidently learn (up to a privacy parameter ε) any individual-level information about the training data. DP-SGD is supported by many open-source libraries [7, 30, 58] and is deployed in state-of-the-art (SOTA) private models with performance increasingly approaching that of non-private models [19].

Subsampling vs. Shuffling in DP-SGD. Since DP-SGD is computationally intensive, practitioners often attempt to optimize it. As DP-SGD processes training data in batches, the standard approach to select batches at each step is *Poisson subsampling*, which requires random access to the entire dataset and can be slow for large datasets [47]. As a result, recent work has often replaced that with *shuffling* the training data and deterministically iterating over fixed-size batches [19, 48, 52]. Additionally, modern ML pipelines (e.g., XLA compilation [45]) are optimized for fixed batch sizes, while Poisson subsampling produces different batch sizes, which can significantly slow down private training [13].

While the privacy analysis of Poisson subsampling is well-studied and admits strong privacy amplification theorems [5, 22], correctly estimating DP-SGD's theoretical guarantees when using shuffling remains an open problem [16, 17, 29]. As a result, it has become common to train models using DP-SGD with shuffling while reporting DP guarantees as if Poisson subsampling were used [19, 39, 47]. (In the rest of the paper, we use DP-SGD (Shuffle) to denote the former DP-SGD (Poisson) for the latter.) This prompts the need to analyze whether this discrepancy affects the actual privacy guarantees of state-of-the-art models using shuffling.

DP Auditing. We build on the concept of DP auditing [21], which consists in estimating and comparing the empirical privacy leakage (denoted as ε_{emp}) from DP mechanisms against their theoretical DP upper bounds (ϵ) [2, 3, 11, 31, 32, 42, 43]. This involves running attacks against the mechanism – e.g., in the case of DP-SGD, inferring whether or not a sample was used to train the model (aka membership inference [49]), and using the adversary's success to estimate ε_{emp}. Finding that $\varepsilon_{emp} > \epsilon$ indicates the presence of DP violations or bugs in the DP algorithm. For the auditing to be effective, it also needs not to be "loose" – i.e., if $\varepsilon_{emp} \ll \epsilon$, the audit may not be exploiting the maximum possible privacy leakage.

Research Problem. In this paper, we present, to the best of our knowledge, the first methodology to audit DP-SGD (Shuffle), aiming to analyze the gap between its empirical privacy leakage and the theoretical guarantees provided by DP-SGD (Poisson). Notably, auditing the former is significantly harder than the latter, and presenting suitable algorithms has remained a largely unaddressed research question (see Section VIII). Unlike for DP-SGD (Poisson), tight theoretical upper bounds are not currently known for DP-SGD (Shuffle) [16, 17], and

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.2026.240597
www.ndss-symposium.org

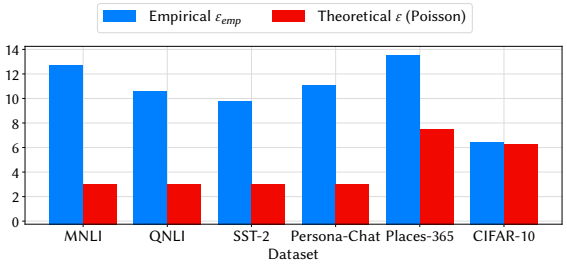

Fig. 1: Largest gaps observed for each dataset between the empirical privacy leakage estimate ε_{emp} and theoretical Poisson guarantees ε when training by shuffling datasets using DP-SGD.

privacy attacks have not been evaluated against it. Therefore, it is unclear whether and how the privacy leakage can be effectively leveraged by the auditing adversary.

Roadmap. We start by auditing a simplified version of DP-SGD, which we denote as Batched Gaussian Mechanism (BGM), using a novel method that builds on likelihood ratio functions. Our BGM variant is non-adaptive and simpler than the Adaptive Batch Linear Query (ABLQ) mechanism studied in prior work [16], enabling us to audit it tightly in a principled way (see Section III). Then, we extend our auditing procedure to DP-SGD (Shuffle) and evaluate the empirical privacy leakage under various adversarial models.

Our experiments highlight a substantial gap (up to $4\times$) between the empirical privacy leakage observed from SOTA models [19, 39] and their claimed theoretical DP guarantees. In Figure 1, we summarize the largest gaps for each dataset we experiment with – e.g., on the MNLI dataset, the state-of-the-art differentially private BERT model from [39] reports a theoretical $\varepsilon = 3$, while our audit results in an empirical $\varepsilon_{emp} = 12.7$ due to shuffling the dataset as opposed to using Poisson sub-sampling as in the theoretical analysis.

However, the gap is not uniform across all parameter settings and threat models. Specifically, for large batch sizes and weak threat models, we find that the gap is much smaller, implying that the relationship between the theoretical Poisson DP guarantees and the actual privacy leakage observed from shuffling is a complex one.

Finally, we adapt our auditing procedures to audit two variations of the shuffling procedure first reported by Ponomareva et al. [47], namely, partial shuffling and batch-then-shuffle. We find them in 2.6% of public code repositories related to non-private ML training and show that these variations yield even larger privacy leakage (up to $10\times$) compared to standard shuffling. For theoretical $\varepsilon = 0.1$, the partial shuffling and batch-then-shuffle procedures result in an empirical privacy leakage of $\varepsilon_{emp} = 0.29$ and $\varepsilon_{emp} = 1.00$, respectively.

Contributions. In short, our work makes several contributions:

- We are the first to audit DP-SGD (Shuffle) and show that the empirical privacy leakage of SOTA models [19, 39] is substantially larger than the theoretical DP guarantees, evaluating the impact of the batch size and the adversarial model on the privacy leakage observed from DP-SGD (Shuffle). In the process, we present novel auditing techniques using likelihood ratio functions.
- We investigate and identify gaps in the privacy leakage from two common variants of the shuffling procedure, namely, partial shuffling and batch-then-shuffle.
- Although we focus on DP-SGD (Shuffle), our auditing framework could be applied to any sampling technique and mechanism, e.g., to identify variations within and estimate privacy leakage for implementations of shuffling itself.
- Our work attests to the impact of shuffling (and its variants) on privacy leakage, calling into question the guarantees claimed by some SOTA models. This has important implications both in terms of privacy (i.e., privacy claims might be overly optimistic) and utility (i.e., hyperparameters tuned to DP-SGD (Shuffle) may not be optimal for DP-SGD (Poisson) and vice versa).

NB: we have made the source code of our auditing procedure publicly available to support reproducibility and encourage further work in this space.[1]

II. Background

A. Differential Privacy (DP)

Definition 1 (Differential Privacy (DP) [24])**.** A randomized mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ is (ε, δ)-differentially private if for any two adjacent datasets $D, D' \in \mathcal{D}$ and $S \subseteq \mathcal{R}$:

$$\Pr[\mathcal{M}(D) \in S] \leq e^{\varepsilon} \Pr[\mathcal{M}(D') \in S] + \delta$$

Two common notions of adjacent datasets are *add/remove* and *edit*. The former corresponds to inserting/deleting a single record from the dataset (hence $|D| = |D'| \pm 1$); the latter to replacing a single record with another ($|D| = |D'|$). Note that the difference in the size of the adjacent datasets limits the applicability of the add/remove adjacency in some settings (e.g., sampling w/o replacement), while guarantees under edit adjacency typically require roughly twice the amount of noise [47] since replacing a record is equivalent to first deleting then adding a record under the add/remove adjacency.

Zero-out Adjacency. To bridge the gap between add/remove and edit adjacency, the "zero-out" adjacency [34] is increasingly used to simplify theoretical privacy analyses [14, 16, 47].

Definition 2 (Zero-out adjacency [34])**.** Let $\mathcal{X}$ be a data domain s.t. special element $\perp \notin \mathcal{X}$ and $\mathcal{X}_{\perp} = \mathcal{X} \cup \{\perp\}$. Datasets $D \in \mathcal{X}^n$ and $D' \in \mathcal{X}_{\perp}^n$ are zero-out adjacent if exactly one record in D is replaced with $\perp$ in D'.

The special $\perp$ record is usually 0 for numerical data. This allows the guarantees under zero-out adjacency to be semantically equivalent to those under add/remove (i.e., no additional noise required) while ensuring that the sizes of the adjacent datasets are equal.

f-DP and trade-off functions. Besides (ε, δ)-DP, there are other formalizations of DP; e.g., f-DP captures the difficulty for any adversary to distinguish between the outputs of a

[1] See https://github.com/spalabucr/audit-shuffle.

mechanism $\mathcal{M}$ on adjacent datasets D and D' using trade-off functions.

Definition 3 (Trade-off function [22])**.** For any two probability distributions P and Q on the same space, the trade-off function $T(P, Q) : [0, 1] \to [0, 1]$ is defined as:

$$T(P, Q)(\alpha) = \inf\{\beta_\phi : \alpha_\phi \leq \alpha\}$$

where the infimum is taken over all (measurable) rejection rules ϕ and α_ϕ and β_ϕ are the type I and II errors corresponding to this rejection rule, respectively.

In theory, the trade-off function characterizes the false positive/negative errors achievable by any adversary aiming to distinguish between P and Q. A mechanism $\mathcal{M}$ is said to satisfy f-DP if for all adjacent datasets D, D': $T(\mathcal{M}(D), \mathcal{M}(D')) \geq f$.

One special case of f-DP is μ-GDP, when the underlying distributions are Gaussian.

Definition 4 (μ-GDP [22])**.** A mechanism $\mathcal{M}$ satisfies μ-GDP if for all adjacent datasets D, D':

$$T(\mathcal{M}(D), \mathcal{M}(D')) \geq \Phi(\Phi^{-1}(1 - \alpha) - \mu),$$

Φ being the standard normal cumulative distribution function.

f-DP is useful, e.g., in the context of auditing (introduced in Section III), as it is equivalent to (ε, δ)-DP for specific trade-off functions and can be used to efficiently compare the empirical power of adversaries with theoretical guarantees, as we do later in the paper.

Privacy Loss Distribution (PLD). The PLD formalism [36] is useful to derive tight theoretical guarantees for DP mechanisms. In recent work [42], PLD has also been used to tightly audit DP-SGD (Poisson) by using the given PLD to estimate the corresponding trade-off function. However, while the PLD for DP-SGD (Poisson) is known, deriving the PLD for DP-SGD (Shuffle) is still an active area of research [16, 17].

B. DP-SGD

Differentially Private Stochastic Gradient Descent (DP-SGD) [1] is a popular algorithm for training machine learning (ML) models with formal privacy guarantees. DP-SGD takes in input a dataset D along with several hyperparameters and proceeds iteratively, processing the dataset in batches and computing the gradients of samples one batch at a time. There are several strategies for sampling a batch from a dataset, such as Poisson subsampling and sampling without replacement. In general, DP-SGD can be defined with an abstract batch sampler $\mathcal{B}$ that takes as input the dataset D and the (expected) batch size B, and outputs batches sampled from the dataset. We report its pseudo-code in Algorithm 4 in Appendix A.

Poisson subsampling. The first implementations of DP-SGD used batches sampled through Poisson subsampling. With this approach, the batch sampler independently samples each record $(x, y) \in D$ with probability $q = B/|D|$ in each batch, where B is the batch size. One major advantage of using Poisson subsampling is substantially reducing the amount of required noise via strong privacy amplification theorems [1, 22]. However, in practice, Poisson subsampling can be quite inefficient. For instance, the dataset cannot be fully loaded in memory when it is too large; thus, one needs to load a random batch in and out from the disk at every step, which is costly [47]. Furthermore, efficient modern ML pipelines (e.g., XLA compilation) require fixed-size batches to fully leverage GPU parallelization [13].

Shuffling. In practice, state-of-the-art DP-SGD implementations often rely on more computationally efficient sampling schemes, such as shuffling, while reporting DP guarantees as though Poisson subsampling was used [19]. In shuffling, the batch sampler randomly permutes the records first, then partitions the dataset into fixed-size batches. Since shuffling is already the standard in non-private training, this also simplifies implementations of DP-SGD by layering DP on top of existing non-private pipelines instead of redesigning and optimizing them from scratch. However, since the privacy guarantees provided by shuffling are not yet fully understood, this can create a discrepancy between the theoretical guarantees reported by state-of-the-art models and the actual privacy leakage, which we study in this work.

Zero-out Adjacency. While shuffling is more compatible with edit adjacency due to the fixed batch sizes, the Poisson scheme is more suitable for add/remove adjancency. Nevertheless, it is possible to derive guarantees for Poisson under the edit adjacency, although this is known to be very cumbersome [5]. Thus, comparing privacy guarantees between the different subsampling schemes is inherently complicated. Overall, zero-out adjacency is increasingly used to analyze the privacy of DP-SGD in practice [14, 34, 47].

Specifically, the $\perp$ record is defined so that the gradient is always zero, i.e., $\forall\theta\ \nabla\ell(\perp; \theta) = \mathbf{0}$ [34]. By doing so, we ensure that DP-SGD (Poisson) under zero-out adjacency is semantically equivalent to DP-SGD (Poisson) under add/remove, which enables us to compare the privacy guarantees of DP-SGD (Poisson) with that of DP-SGD (Shuffle) under a common adjacency notion.

III. DP Auditing

Privacy auditing broadly denotes the process of empirically estimating the privacy leakage from an algorithm. In DP, this involves running experiments to estimate the empirical privacy guarantees (ε_{emp}) and compare them to the theoretical guarantees (ε). For simplicity, we assume the empirical guarantees are derived for the same δ as the theoretical (ε, δ)-DP; thus, we are technically comparing $(\varepsilon_{emp}, \delta)$ to (ε, δ), but leave out δ to ease presentation. If one finds that $\varepsilon_{emp} > \varepsilon$, the mechanism leaks more privacy than expected. As mentioned earlier, if $\varepsilon_{emp} \ll \varepsilon$, the audit is not *tight*, i.e., the theoretical bounds are overly conservative or there may be substantial room for improvement to the privacy estimation procedure.

Implementing DP algorithms correctly is challenging [10, 33] and bugs in DP-SGD implementations have resulted in significantly degraded protections [11, 33] and realistic privacy

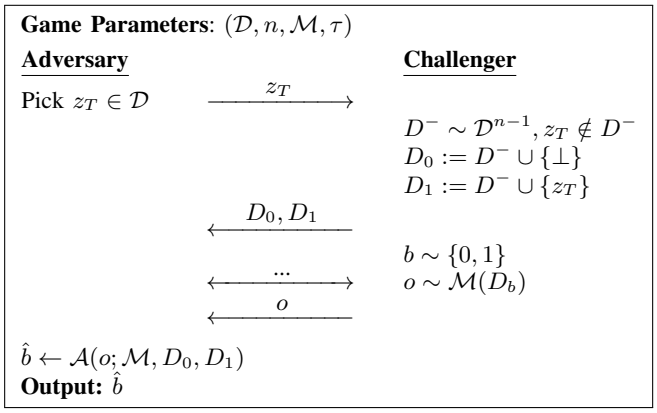

Fig. 2: Distinguishability Game between an Adversary and a Challenger for zero-out DP given data distribution $\mathcal{D}$ of size n, the mechanism $\mathcal{M}$, and a decision threshold τ.

attacks [20]. DP auditing can be used to detect these bugs [4, 42, 53] and/or evaluate the optimality of the attacks [4, 43]. Moreover, empirical guarantees are also useful for estimating the privacy loss in settings where tight theoretical guarantees are not currently known, e.g., when the adversary does not have access to intermediate updates [2, 3].

In the rest of this section, we introduce a general DP auditing procedure using an adversary and a distinguishing game. The attack's success can then be converted into the lower-bound empirical privacy leakage estimate ε_{emp}.

A. The Distinguishing Game

In Figure 2, we present the standard distinguishing game [42], involving an Adversary and a Challenger, adapted to audit the zero-out DP guarantees of a mechanism $\mathcal{M}$. The Challenger runs the mechanism on a randomly chosen dataset, and the Adversary aims to determine the dataset used by the Challenger based on the mechanism's output.

In each game, the Adversary picks a single target record z_T from the data domain and sends it to the Challenger.[2] The latter constructs adjacent datasets D_0 and D_1 by appending the zero-out record $\perp$ and the target record z_T, respectively, to a randomly sampled dataset with $n-1$ records as done in prior work [2, 42, 46, 50]. Then, the Adversary is given access to the randomly sampled adjacent datasets D_0 and D_1. Next, the Challenger runs $\mathcal{M}$ on dataset D_b for a random $b \in \{0, 1\}$, and the Adversary wins if they correctly guess $\hat{b} = b$.

Note that $\mathcal{M}$ can be a complex algorithm with many implementation details that the theoretical privacy analysis does not rely on; thus, depending on the threat model considered and the adversarial capabilities, the adversary may be given access to specific internals of $\mathcal{M}$ as discussed in Section V-A, along with M's output. In theory, this can take any form, e.g., scalar $o \in \mathbb{R}$, vector $o \in \mathbb{R}^d$, or even have arbitrary domain $o \in \mathcal{Y}$. However, it can be difficult to consider and design decision functions around values in arbitrary domains. Hence, a common strategy [31, 43] is for the Adversary to define a distinguishing function $\mathcal{A}$ that assigns a scalar "score" to the output representing the Adversary's confidence that the output is drawn from processing D_1. This score is then thresholded to produce a guess $\hat{b} = 1$ if the Adversary determines that $o \sim \mathcal{M}(D_1)$ and 0 otherwise.

[2]We are abusing notation by define $\mathcal{D}$ as both the data *distribution* and *domain*, although it is clear from the context.

B. Estimating ε_{emp}

We estimate the empirical privacy leakage ε_{emp} by running the distinguishing game multiple times, where only a single record is inserted in each run. We compute the false positive rate (FPR) α and the false negative rate (FNR) β across multiple games and derive (statistically valid) upper bounds $\overline{\alpha}$ and $\overline{\beta}$ using Clopper-Pearson confidence intervals (CIs) to quantify the confidence in our privacy loss estimation [31, 42]. We then calculate the empirical lower-bound privacy guarantee ε_{emp} from $\overline{\alpha}$ and $\overline{\beta}$ using the (ε, δ)-DP definition as follows.

While recent DP-SGD audits have presented "auditing in one run" methods [41, 50, 55], these do not yield sufficiently tight DP-SGD audits (even under powerful threat models), and thus we choose to audit using multiple runs. Moreover, prior work [41, 42, 55] has proposed alternative auditing techniques for DP-SGD (Poisson) using f-DP or PLD; however, since tighter f-DP guarantees for DP-SGD (Shuffle) are currently unknown [16], we stick to the (ε, δ)-DP definition.

Auditing using (ε, δ)-DP. For any given (ε, δ)-DP mechanism, the possible false positive rates (α) and false negative rates (β) attainable by any adversary are known to be the following privacy region [35]:

$$\mathcal{R}(\varepsilon, \delta) = \{(\alpha, \beta) | \alpha + e^{\varepsilon}\beta \geq 1 - \delta \wedge e^{\varepsilon}\alpha + \beta \geq 1 - \delta \wedge \alpha + e^{\varepsilon}\beta \leq e^{\varepsilon} + \delta \wedge e^{\varepsilon}\alpha + \beta \leq e^{\varepsilon} + \delta\} \quad (1)$$

Therefore, given upper bounds $\overline{\alpha}$ and $\overline{\beta}$, the empirical lower bound can be calculated as:

$$\varepsilon_{emp} = \max\left\{\ln\left(\frac{1-\overline{\alpha}-\delta}{\overline{\beta}}\right), \ln\left(\frac{1-\overline{\beta}-\delta}{\overline{\alpha}}\right), 0\right\} \quad (2)$$

While Bayesian intervals [59] can improve the privacy estimation's tightness, they are also more computationally expensive to derive and may not always be statistically sound [42]. Thus, we stick to Clopper-Pearson CIs. (In Section VI-B, we will also evaluate the impact of using CIs on the empirical privacy leakage estimation).

For simplicity, we follow standard practice [31, 42, 50, 59] and fix the the value of $\delta = 10^{-5}$ throughout all experiments and only estimate the ε_{emp} guarantee at this δ value.

C. The Distinguishing Function

As mentioned, DP auditing involves an adversary distinguishing between observations from $\mathcal{M}(D)$ and $\mathcal{M}(D')$. For a mechanism $\mathcal{M} : \mathcal{X} \to \mathcal{Y}$, this can be any function of the form $\mathcal{A} : \mathcal{Y} \to \{0, 1\}$ where $\mathcal{A}(o) = 1$ (or 0) represents that the adversary predicts that the observation o is drawn from $\mathcal{M}(D)$ (or $\mathcal{M}(D')$).

Scalar Score. In theory, $\mathcal{A}$ is equivalent to the rejection rule from the definition of f-DP, which represents the adversary performing a hypothesis test on whether to reject that

$o \sim \mathcal{M}(D')$. However, in previous work [31, 32, 42, 43], the adversary assigns a scalar "score" to each output, which is then thresholded to form the distinguishing function—e.g., Jagielski et al. [31] and Nasr et al. [42] use, respectively, the loss function and dot product. This is because the raw outputs of DP-SGD tend to be high-dimensional vectors where distances can be difficult to interpret, thus making distinguishing functions hard to design.

On the other hand, using a scalar score, the adversary can first represent the *confidence* that the observation was drawn from $\mathcal{M}(D)$ instead of $\mathcal{M}(D')$. Then, the score can be thresholded to produce a prediction easily, i.e., all observations with score $\geq \tau$ are labeled as drawn from $\mathcal{M}(D)$ and from $\mathcal{M}(D')$ otherwise. Furthermore, this threshold τ can also be adjusted to produce not only a single FPR/FNR pair but an FPR-FNR curve, i.e., an empirical *trade-off curve*, which can also be compared with the claimed theoretical trade-off function when auditing.

Choosing the threshold. When auditing using scalar scores, choosing the threshold τ from an independent set of observations is the easiest method to ensure a technically valid lower bound for the empirical privacy leakage estimate ε_{emp} derived this way. However, it is standard to report the maximum ε_{emp} for the optimal threshold [3, 11, 31, 42, 43, 46, 59] and we do so too, making the process more efficient by first sorting the scalar scores. Here, we substantially improve the efficiency of this procedure by first sorting the scores in increasing order, as reported in Algorithm 1. This enables us to find the optimal threshold from, potentially, billions of observations without incurring a prohibitive computational cost. For simplicity, we abstract the details of choosing an appropriate threshold and designing a distinguishing function. We refer to the empirical estimation procedure at significance level α and privacy parameter δ from scores $\mathcal{S}$ and $\mathcal{S}'$ derived from $\mathcal{M}(D)$ and $\mathcal{M}(D')$, respectively as EstimateEps$(\mathcal{S}, \mathcal{S}', \alpha, \delta)$.

Algorithm 1 Estimating ε_{emp} from scores

Require: Scores from $\mathcal{M}(D)$, $\mathcal{S}$. Scores from $\mathcal{M}(D')$, $\mathcal{S}'$. Significance level, α. Privacy parameter, δ.
▷ Assume $|\mathcal{S}| = |\mathcal{S}'|$.
$R \leftarrow |\mathcal{S}|$
▷ Initial FPR and FNR for classifying all scores as $s \sim \mathcal{M}(D)$.
FPR $\leftarrow R$
FNR $\leftarrow 0$
for $\tau \in$ sort_increasing$(\mathcal{S} \cup \mathcal{S}')$ **do**
▷ Calculate FPR and FNR for classifying score as $s \sim \mathcal{M}(D)$ if $s > \tau$.
  if $\tau \in S$ **then**
    FNR $\leftarrow$ FNR $+ 1$
  else
    FPR $\leftarrow$ FPR $- 1$
  end if
  ▷ Calculate confidence intervals for FPR and FNR.
  $\overline{\text{FPR}} \leftarrow$ Clopper-Pearson(FPR, R, α)
  $\overline{\text{FNR}} \leftarrow$ Clopper-Pearson(FNR, R, α)
  ▷ Estimate ε_{emp} from Equation 2
  $\varepsilon_{emp}[\tau] = \max\left\{\ln\left(\frac{1-\overline{\text{FPR}}-\delta}{\overline{\text{FNR}}}\right), \ln\left(\frac{1-\overline{\text{FNR}}-\delta}{\overline{\text{FPR}}}\right), 0\right\}$
end for
return $\max_\tau \varepsilon_{emp}[\tau]$.

IV. Auditing the Batched Gaussian Mechanism

Before auditing DP-SGD (Shuffle), we first turn to the Batched Gaussian Mechanism (BGM) (see Algorithm 5 in Appendix A), a heavily *simplified* version of DP-SGD adapted from [16]. We do so to develop principled, (close to) tight auditing techniques for shuffling (under an idealized setting).

Like DP-SGD, BGM also proceeds iteratively, sampling batches according to some batch sampler $\mathcal{B}$. However, instead of calculating gradients, the inputs are simply aggregated together with noise at each batch, and the noisy aggregated values are released for each batch across all epochs. Moreover, unlike DP-SGD, it is *non-adaptive*, i.e., the outputs of previous iterations do not affect the current one.

While BGM can be instantiated with any batch sampler, we use the *shuffle* batch sampler, which is commonly used in ML training. We bound the inputs, i.e. $\forall i\ x_i \in [-1, +1]$, so that the mechanism satisfies (ε, δ)-DP. When the number of epochs is 1, Chua et al. [16] use the adjacent datasets $D = (+1, -1, ..., -1)$ and $D' = (\perp, -1, ..., -1)$, setting $\perp = 0$ in their lower bound privacy analysis. Specifically, they conjecture, but do not prove, that these are the worst-case adjacent datasets for the shuffle batch sampler. Note that this is different from the proven worst-case datasets $D = (+1, 0, ..., 0)$ and $D' = (\perp, 0, ..., 0)$ for the Poisson batch sampler [60]. Specifically, in the shuffle setting, the differing sample in the conjectured worst-case datasets is equal in magnitude but opposite in direction to the other samples, unlike in the Poisson setting. Finally, in a single epoch and for batch size B, the outputs of the mechanism on the adjacent datasets are $(-B, ..., -B+2, ..., -B) + \mathcal{N}(0, \sigma^2\mathbb{I})$ and $(-B, ..., -B+1, ..., -B) + \mathcal{N}(0, \sigma^2\mathbb{I})$ for D and D', respectively. Since the inputs are shuffled, the "$-B+2$" and "$-B+1$" values can appear in any batch.

We stress that BGM is a "hypothetical" algorithm, i.e., it is not used/associated with common data distributions or training datasets. Therefore, when auditing it using the Distinguishability Game, we take the data distribution to be the "pathological" distribution, which always outputs '-1.' Then, the adversary always chooses '+1' as the target record z_T.

A. From Single to Multiple Epochs

Single Epoch. To audit a single epoch, an adversary has to distinguish between $(\tilde{g}_1, ..., \tilde{g}_T) \sim \text{BGM}(D)$ and $(\tilde{g}'_1, ..., \tilde{g}'_T) \sim \text{BGM}(D')$. As mentioned, prior DP-SGD audits [31, 32, 42, 43] typically compute a natural "score" from each thresholded observation. However, in the context of shuffling, there is no known score to distinguish between the outputs.

In this work, we draw from the Neyman-Pearson [44] lemma, which states that the optimal way to distinguish between two distributions is by thresholding the output of the *likelihood ratio* function. Specifically, the likelihood ratio function computes the ratio of probabilities that the outputs are from BGM(D) or BGM(D'). Here, the probability that

$(\tilde{g}_1, ..., \tilde{g}_T)$ is from BGM(D) (or BGM(D')) can be split into T cases depending on which batch the target record $+1$ (or $\perp$) appears in. To that end, we calculate the likelihood ratio for the adjacent datasets $D = (+1, -1, ..., -1)$ and $D' = (\perp, -1, ..., -1)$ as follows (we let $\tilde{\mathbf{g}} = (\tilde{g}_1, ..., \tilde{g}_T)$):

$$\Lambda(\tilde{\mathbf{g}}) = \frac{\Pr[\tilde{\mathbf{g}}|\text{BGM}(D)]}{\Pr[\tilde{\mathbf{g}}|\text{BGM}(D')]} = \frac{\sum_t T^{-1} \Pr[\tilde{\mathbf{g}}|\text{BGM}(D) \wedge +1 \text{ is in batch } t]}{\sum_t T^{-1} \Pr[\tilde{\mathbf{g}}|\text{BGM}(D') \wedge \perp \text{ is in batch } t]} =$$
$$= \frac{\sum_t \Pr[\tilde{g}_t|\mathcal{N}(-B+2, \sigma^2)] \prod_{t' \neq t} \Pr[\tilde{g}_{t'}|\mathcal{N}(-B, \sigma^2)]}{\sum_t \Pr[\tilde{g}_t|\mathcal{N}(-B+1, \sigma^2)] \prod_{t' \neq t} \Pr[\tilde{g}_{t'}|\mathcal{N}(-B, \sigma^2)]}$$

Note that for arbitrary adjacent datasets where all records are different, $\Lambda(\cdot)$ would be computationally intractable to compute since we would need to account for all possible permutations induced by shuffling. However, since [16]'s conjecture assumes that all records except one are identical, this reduces the number of "cases" to be considered to T, thus making $\Lambda(\cdot)$ tractable.

Multiple Epochs. Although [16] only considers one set of batches (i.e., corresponding to a single "epoch" of training), we extend to multiple epochs as training private models typically involves multiple epochs. We rely on the fact that the batch sampling and aggregation are done *independently* across multiple epochs; thus, the likelihood across multiple epochs is the product of the likelihoods of each epoch, i.e., letting $\mathcal{G} = \begin{bmatrix} -\tilde{\mathbf{g}}^1- \\ \vdots \\ -\tilde{\mathbf{g}}^E- \end{bmatrix}$, we define the likelihood ratio as:

$$\Lambda^{\text{BGM}}(\mathcal{G}) = \prod_{i=1}^{E} \Lambda(\tilde{\mathbf{g}}^i) =$$
$$= \prod_{i=1}^{E} \frac{\sum_t \Pr[\tilde{g}^i_t|\mathcal{N}(-B+2, \sigma^2)] \prod_{t' \neq t} \Pr[\tilde{g}^i_{t'}|\mathcal{N}(-B, \sigma^2)]}{\sum_t \Pr[\tilde{g}^i_t|\mathcal{N}(-B+1, \sigma^2)] \prod_{t' \neq t} \Pr[\tilde{g}^i_{t'}|\mathcal{N}(-B, \sigma^2)]}$$

Overall, our experimental evaluation (presented below) shows that auditing using likelihood ratios is very effective at estimating the privacy leakage from the Batched Gaussian Mechanism. In this idealized setting, we observe a substantial gap (up to 4×) between the privacy leakage observed empirically and the theoretical guarantees from the Poisson subsampling analysis (recall that there are no known guarantees for shuffling). This further motivates us to investigate whether this gap is also observable in real-world settings using DP-SGD.

B. Experiments

Varying Batch Size and Noise Multiplier. We start by fixing both the number of epochs and the batch size (B) to 1. We vary the *number of steps* for a single epoch, T, by varying the dataset size and can then compute the theoretical Poisson analysis ε using the privacy loss random variable (PRV) accountant for a given T and noise multiplier σ. Furthermore, to assess computational complexity, we audit each setting over a varying number of game runs, each yielding a single observation. For context, we also consider the lower bound theoretical ε previously calculated by Chua et al. [16].

In Figure 3, we plot the empirical ε_{emp} observed when auditing the Batched Gaussian Mechanism with the numbers

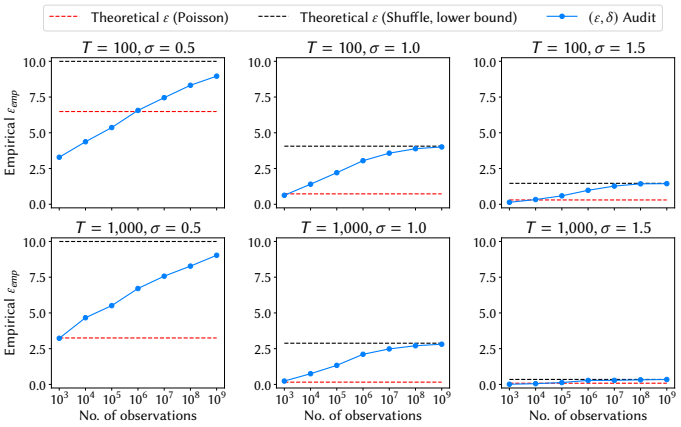

Fig. 3: Auditing BGM for various number of steps T and noise multipliers σ. Theoretical ε (Poisson) represents the theoretical guarantees if Poisson subsampling had been used, while Theoretical ε (Shuffle, lower bound) denotes the lower bounds from [16].

of steps T at 100 and 1,000 and noise multipliers σ at $0.5, 1.0, 1.5$. For several parameters, the empirical privacy leakage is substantially larger than the theoretical upper bounds from the Poisson analysis. Specifically, with $T = 100$, we obtain $\varepsilon_{emp} = 8.96, 4.01$, and 1.44, respectively, for $\sigma = 0.5, 1.0, 1.5$ against theoretical upper bounds suggesting only $\varepsilon = 6.49, 0.73, 0.30$ (over 10^9 observations).

This gap varies with different numbers of steps taken in each epoch (T) and different noise multipliers (σ). More precisely, it is substantially larger for larger T (corresponding to small batch sizes) and smaller σ. For instance, the gap between ε and ε_{emp} is 5.78 for $T = 1000$, $\sigma = 0.5$ vs. 1.14 for $T = 100$, $\sigma = 1.5$. However, larger numbers of steps may not always result in larger empirical privacy leakage estimates either. At $\sigma = 1.5$, for $T = 100$ and 1000, we find $\varepsilon_{emp} = 1.44$ and $\varepsilon_{emp} = 0.34$, respectively.

Since the privacy loss distribution (PLD) used in [42] is not currently known for DP-SGD (Shuffle), recall that we are auditing using the (ε, δ)-DP definition; as a result, we require a large number of observations for the ε_{emp} estimates to converge, especially when σ is small. For $\sigma \geq 1.0$, at $T = 100$, ε_{emp} converges after 10^8 observations, but for $\sigma = 0.5$, it requires more than 10^9 observations (we do not explore beyond 10^9 due to computational constraints).

As expected, given enough observations, our audits match but do not exceed the theoretical lower bounds calculated by Chua et al. [16]. This not only confirms that our auditing procedure is effective but also that it might be possible to convert the *lower-bound privacy analysis* from [16] into an *upper-bound privacy guarantee*. However, proving upper-bound privacy guarantees might not be trivial and is beyond our scope—thus, we leave it to future work.

Exploring parameter settings. Next, we audit using parameters from state-of-the-art differentially private models. Specifically, we audit BGM in the settings used in [19] and [39] to train private image classification models and large language

Paper	Dataset	Size	σ	T	E	ε	ε_{emp}	Gap
[19]	Places-365	1.8M	1.00	440	509	7.53	13.54	1.80×
[19]	CIFAR-10	60K	3.00	11	168	6.24	6.39	1.02×
[39]	SST-2	60K	0.79	117	10	3.00	9.80	3.27×
[39]	QNLI	100K	0.87	195	30	3.00	10.60	3.53×
[39]	MNLI	400K	0.73	781	50	3.00	12.73	4.25×
[39]	Persona-Chat	400K	0.82	254	30	2.99	11.08	3.70×

TABLE I: Parameter settings used in prior work where the empirical privacy leakage (ε_{emp}) observed from shuffling is appreciably larger than the theoretical ε from Poisson sampling.

models, respectively[3]. Note that De et al. [19] explicitly state they train their models with shuffling but report DP guarantees as though Poisson sampling was used. Whereas, Li et al. [39] do not mention using shuffling, but a review of their codebase shows they use the default Pytorch implementation of `DataLoader`, which shuffles the dataset.

In total, we audit all 101 combinations of parameters (i.e., noise multiplier, σ, number of steps per epoch, T, and number of epochs E) extracted from [19] and [39] using 10^7 observations. Overall, we find a substantial gap between the empirical privacy leakage under shuffling and the theoretical Poisson analysis in two-thirds of the settings studied. In Table I, we report the maximum gaps between ε_{emp} and ε, over several datasets, for parameter settings used in prior work. The largest gap (4.25×) occurs with large language models, probably owing to the small noise multiplier used to optimize utility.

Overall, these experiments indicate that, under the Batched Gaussian Mechanism's ideal conditions, training with shuffling instead of Poisson subsampling incurs substantial privacy leakage, also in settings commonly used for training state-of-the-art models. This further motivates assessing the validity of the DP guarantees reported by state-of-the-art models.

C. Takeaways

As mentioned earlier, we experiment with Batched Gaussian Mechanism and study the impact of various parameter settings (noise scale, batch size, and number of epochs) on the empirical privacy leakage observed. Our experiments show that the empirical privacy leakage from shuffling can be substantially larger than the theoretical guarantees given by the Poisson subsampling analysis.

Crucially, this gap is prominent in the parameter settings used to train SOTA models in prior work [19, 39] as well. For instance, De et al. [19] reportedly use shuffling to fine-tune an NF-ResNet-50 model on the Places-365 dataset at a theoretical $\varepsilon = 7.53$, but our audits show that the actual privacy leakage is almost double ($\varepsilon_{emp} = 13.59$).

V. Auditing DP-SGD (Shuffle)

In this section, we present our novel auditing procedure for DP-SGD (Shuffle). We start by defining various threat models specific to the shuffling setting and then present our algorithm.

[3]We audit the different parameter settings using our own shuffling implementation instead of auditing existing implementations both for computational efficiency and to freely vary the hyperparameters (e.g., batch size).

A. Adversarial Modeling

Although DP-SGD (Shuffle)'s structure is very similar to that of BGM, there are practical considerations that can affect the empirical privacy estimates. For instance, since the inputs to the mechanism are directly aggregated and all intermediate noisy aggregates are released, adversaries for the BGM are, by default, given as much power as the theoretical worst-case DP adversary. By contrast, DP-SGD first calculates the gradient of each input before aggregating them and may only output the final trained model, thus making it unclear which aspects of the mechanism the adversary is given access to.

Auditing with a `Worst-Case` Adversary. We start by considering the worst-case adversary, assuming they can insert the gradients of all records in the dataset. Although this may destroy model utility, recall that the main goal of DP auditing is to verify that the DP bounds are correct, and DP is a worst-case guarantee that should hold against the most powerful adversary. In fact, the privacy analysis of DP-SGD assumes that the adversary can indeed insert arbitrary gradients for all records [1].

Overall, adversaries used in the DP auditing literature are usually given strong capabilities, including, e.g., (active) white-box access to the model [42, 43]. More precisely, DP-SGD audits [4, 11, 41–43, 46, 50, 55] typically consider a "white-box with gradient canaries" setting where not only can the adversary choose the target record $(\hat{x}, \hat{y})$, but they also have access to the model parameters at each update step and can arbitrarily insert gradients of samples (i.e., gradient canary). Our work also operates in this setting.

Auditing with a `Target-Canary` Adversary. Prior DP-SGD audits [4, 11, 41–43, 46, 50, 55] have also experimented with adversaries that can only insert the gradient of the target record, obtaining relatively tight audits for DP-SGD (Poisson). In this setting, all records except the target have natural gradients (i.e., not adversarially crafted). While we expect models audited using these adversaries to have the same utility as production models (the canary gradient has minimal impact on it), our experiments show the resulting audits to be relatively loose. As a result, we only report experimental results in the context of analyzing the impact of varying adversarial capabilities on the tightness of the audit (as also done in prior work [3, 31, 42, 43, 50]).

Auditing with a `Partially-Informed` Adversary. Theoretically, the worst-case adversary assumed by the DP guarantees has access to all aspects of the algorithm not involved in the randomness for noise addition and batch selection, i.e., it can modify the gradients of *all* samples. Nevertheless, we believe it is interesting and useful to relax this assumption and introduce the so-called `Partially-Informed` adversary, which can only modify *selected* samples' gradients. Specifically, this adversary only inserts the gradients of the target record and the final record *in each batch*. We are motivated to do so as one of the reasons why `Target-Canary` audits are loose could stem from the *bias* introduced by the other samples in each batch, which our likelihood-based auditing procedure could

be sensitive to. Therefore, by leaving the remaining $B-1$ samples in each batch untouched, we can precisely study the impact of the *bias* introduced by other samples on our auditing procedure. Similar to the `Worst-Case`, the adversary can also insert opposing gradients, thus, making the batches containing the target or zero-out record more obvious. Overall, models audited under this threat model should still maintain similar utility as production models, while we expect them to result in tighter audits.

Auditing with Strong Adversaries. Although our adversarial models consider relatively strong, gradient-crafting adversaries, arguably, this does not diminish the value and impact of our analysis. First of all, as we discuss in Section VIII, prior work has considered similarly strong adversaries: for instance, in the context of DP-SGD (Poisson), Nasr et al. [43] rely on adversaries that can craft arbitrarily malicious datasets (i.e., "Pathological Dataset"). In other words, our Worst-Case adversary can be modeled with the Pathological Dataset adversary; thus, our threat model does not involve adversaries stronger than those used in prior auditing work (see Appendix C).

Overall, it is common to instantiate audits with different adversarial models and evaluate the resulting impact on their tightness [4, 42, 43]. We follow a similar strategy and consider several adversarial models that are specifically catered to DP-SGD (Shuffle); this also allows us to evaluate the impact of adversarial capabilities on different aspects of DP-SGD (Shuffle) algorithms in terms of privacy leakage.

Perhaps more importantly, being a worst-case definition, DP is meant to provide formal guarantees that hold also against worst-case adversaries. One of the main goals of DP auditing is to verify that empirical leakage does not exceed these (theoretical) DP guarantees. In our case, in particular, the main research question we focus on is whether or not privacy discrepancies emerge from designing DP-SGD algorithms that use shuffling but reporting guarantees as if Poisson subsampling was used. Therefore, the adversaries we experiment with should be seen as part of our auditing toolkit, more than in the context of how "realistic" it would be to instantiate them.

B. Auditing Procedure

In Algorithm 2, we outline the DP-SGD algorithm [1] with the modifications needed to audit with `Target-Canary`, `Partially-Informed`, and `Worst-Case` adversaries. Lines 7-8 correspond to enforcing that the $\perp$ record has a $\mathbf{0}$ gradient, which is needed for auditing with zero-out adjacent datasets.

Then, lines 9-10 are standard steps for active white-box adversaries that insert the gradient of the target record. Lines 11-12 correspond to inserting the gradient of the final record in each batch, which only occurs with `Partially-Informed` and `Worst-Case` (but not `Target-Canary`). Specifically, the adversary inserts the canary gradient in the opposite direction for each batch that does not contain both the target and zero-out records. This closely resembles the conjectured worst-case datasets by Chua et al. [16] as the target record's gradient has an equivalent magnitude but opposite direction to the final records in all other batches, which makes audits in this model tighter. This does not break or change the underlying DP guarantees of DP-SGD (Shuffle) as this step is applied regardless of whether the target or zero-out record is present in the dataset and thus applies to D and D' equally.

In line 15, the `Worst-Case` adversary additionally zeroes the gradients of all other samples, effectively removing the bias from the other samples. Finally, all adversaries compute the dot product between the privatized and canary gradient as "outputs" (line 21) and release the full matrix of "outputs" across all batches and epochs (line 24). Similar to BGM, the adversary computes and thresholds the likelihood ratio to calculate empirical privacy leakage estimates.

In our auditing procedure, the adversary does not have access to the underlying batch sampler $\mathcal{B}$, but only to the output matrix $\mathcal{O}$. We do so as we focus on determining the impact of alternate subsampling schemes on the empirical privacy leakage. This allows us not only to assess whether the adversary can leak more privacy without knowing the specifications of the batch sampler but also to detect bugs within the shuffling implementations (see Section VII).

Overall, our auditing procedure assumes an adversary that

Algorithm 2 The DP-SGD algorithm with the modifications we make to audit DP-SGD (Shuffle) with different adversaries. Modifications for `Target-Canary`, `Partially-Informed`, and `Worst-Case` are reported in red. Additional modifications for `Partially-Informed` and `Worst-Case` only are in blue and for `Worst-Case` only in orange.

Require: Dataset, D. Epochs, E. Batch Size, B. Learning rate, η. Batch sampler, $\mathcal{B}$. Loss function, ℓ. Initial model parameters, θ_0. Noise multiplier, σ. Clipping norm, C. Target record, $(\hat{x}, \hat{y})$. Canary gradient, $\hat{g}$. Zero-out record, $(x_\perp, y_\perp)$.

1: $T \leftarrow |D|/B$
2: **for** $i \in [E]$ **do**
3: $\quad\theta_1^i \leftarrow \theta_T^{i-1}$
4: $\quad$Sample batches $B_1, ..., B_T \leftarrow \mathcal{B}(D, B)$
5: $\quad$**for** $t \in [T]$ **do**
6: $\quad\quad$**for** $(x_j, y_j) \in B_t$ **do**
7: $\quad\quad\quad$**if** $(x_j, y_j) = (x_\perp, y_\perp)$ **then**
8: $\quad\quad\quad\quad g_j \leftarrow \mathbf{0}$
9: $\quad\quad\quad$**else if** $(x_j, y_j) = (\hat{x}, \hat{y})$ **then**
10: $\quad\quad\quad\quad g_j \leftarrow \hat{g}$
11: $\quad\quad\quad$**else if** $j = B - 1 \wedge (\hat{x}, \hat{y}), (x_\perp, y_\perp) \notin B_t$ **then**
12: $\quad\quad\quad\quad g_j \leftarrow -\hat{g}$
13: $\quad\quad\quad$**else**
14: $\quad\quad\quad\quad g_j \leftarrow \nabla\ell((x_j, y_j); \theta_t^i)$
15: $\quad\quad\quad\quad g_j \leftarrow 0$
16: $\quad\quad\quad$**end if**
17: $\quad\quad\quad\bar{g}_j \leftarrow g_j / \max\left(1, \frac{||g_j||_2}{C}\right)$
18: $\quad\quad$**end for**
19: $\quad\quad\tilde{g} \leftarrow \frac{1}{B}\left(\sum_j \bar{g}_j + \mathcal{N}(0, C^2\sigma^2\mathbb{I})\right)$
20: $\quad\quad\theta_{t+1}^i \leftarrow \theta_t^i - \eta\tilde{g}$
21: $\quad\quad o_t^i \leftarrow \langle \tilde{g}, \hat{g} \rangle$
22: $\quad$**end for**
23: **end for**
24: **return** $\mathcal{O} = \begin{bmatrix} o_1^1 & \dots & o_T^1 \\ \vdots & \ddots & \vdots \\ o_1^E & \dots & o_T^E \end{bmatrix}$

can access all parts of training other than the clipping, noise addition, and batch selection steps and is only slightly weaker than the adversary assumed by DP-SGD's privacy analysis (i.e., the optimal adversary can also choose a pathological dataset D and has access to the batch sampler).

C. Computing Likelihood Ratios

The last step in the audit is for the adversary to calculate the score assigned to the mechanism's output using the Neyman-Pearson lemma. While this is similar to the BGM setting, the likelihood ratios differ depending on the threat model. Mainly, $o_t^i = \langle \tilde{g}, \hat{g} \rangle \approx 0$ if $\hat{g}$ or $-\hat{g}$ were not inserted by the adversary in Steps 10 or 12 and $o_t^i \approx +1$ and ≈ -1, respectively, otherwise. For each threat model, we calculate $\Lambda(\mathcal{O})$ as follows.

- `Target-Canary` (Λ^{TC}):

$$\prod_{i=1}^{E} \frac{\sum_t \Pr[o_t^i | \mathcal{N}(+1, \sigma^2)] \prod_{t' \neq t} \Pr[o_{t'}^i | \mathcal{N}(0, \sigma^2)]}{\prod_t \Pr[o_t^i | \mathcal{N}(0, \sigma^2)]}$$

- `Partially-Informed` (Λ^{PI}):

$$\prod_{i=1}^{E} \frac{\sum_t \Pr[o_t^i | \mathcal{N}(+1, \sigma^2)] \prod_{t' \neq t} \Pr[o_{t'}^i | \mathcal{N}(-1, \sigma^2)]}{\sum_t \Pr[o_t^i | \mathcal{N}(0, \sigma^2)] \prod_{t' \neq t} \Pr[o_{t'}^i | \mathcal{N}(-1, \sigma^2)]}$$

- `Worst-Case` (Λ^{WC}):

$$\prod_{i=1}^{E} \frac{\sum_t \Pr[o_t^i | \mathcal{N}(-B+2, \sigma^2)] \prod_{t' \neq t} \Pr[o_{t'}^i | \mathcal{N}(-B, \sigma^2)]}{\sum_t \Pr[o_t^i | \mathcal{N}(-B+1, \sigma^2)] \prod_{t' \neq t} \Pr[o_{t'}^i | \mathcal{N}(-B, \sigma^2)]}$$

VI. Experimental Evaluation

We now present an experimental evaluation of our auditing techniques, aiming to compare the empirical privacy leakage observed from mechanisms that use shuffling (ε_{emp}) with the upper-bound privacy leakage guaranteed by the analysis using Poisson subsampling (ϵ).

A. Experimental Overview

Datasets. We use three datasets commonly used in the DP auditing literature: FMNIST [56], CIFAR-10 [37], and Purchase-100 (P100) [49]. Due to computational constraints, we only take samples corresponding to two labels from FMNIST and CIFAR-10 and downsample all datasets only to include 10,000 samples, as also done in prior work [31]. Therefore, our FMNIST dataset contains 10,000 28x28 grayscale images from one of two classes ('T-shirt' and 'Trouser'), CIFAR-10 contains 10,000 3x32x32 RGB images from one of two classes ('Airplane' and 'Automobile'), and P100 consists of 10,000 records with 600 binary features from one of 100 classes.

Models. For CIFAR-10, we train a moderate-sized Convolutional Neural Network (CNN) drawn from prior work [23]. For FMNIST and P100, we train a small LeNet and an MLP model, respectively. We refer to Appendix B for more details on the model architectures.

Experimental Testbed. We run all experiments on a cluster using 4 NVIDIA A100 GPUs, 64 CPU cores, and 100GB RAM. Auditing a single model using 10^6 observations took 38.8, 17.1, and 5.70 hours for the shallow CNN, LeNet, and MLP models, respectively.

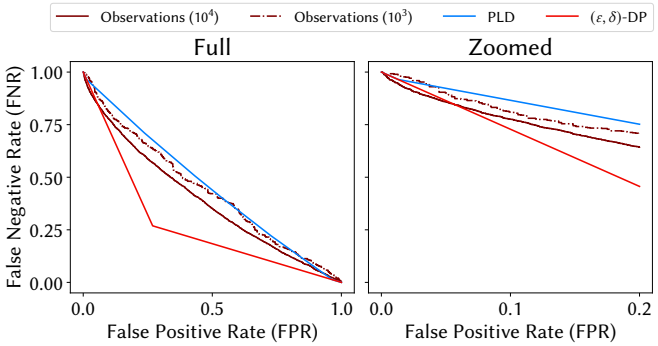

Fig. 4: Comparison of tradeoff curves for auditing DP-SGD (Shuffle) (95% Clopper-Pearson upper bound) at theoretical $\varepsilon = 1.0$ vs. the corresponding theoretical tradeoff curves from (ε, δ)-DP and from the PLD analysis of DP-SGD (Poisson). The plot on the right is zoomed in on $0 \leq \text{FPR} \leq 0.2$.

Metrics and Parameters. For all experiments, we report lower bounds with 95% confidence (Clopper-Pearson [18]) along with the mean and standard deviation values of ε_{emp} over five independent runs. For simplicity, we set $\delta = 10^{-5}$, gradient clipping norm to $C = 1.0$, and choose the learning rate η by hyperparameter tuning from a logarithmic scale.

B. Auditing DP-SGD (Shuffle) with Partially-Informed

Next, we audit DP-SGD (Shuffle) in the `Partially-Informed` model described in Section V-A. We do so because we expect `Partially-Informed` to produce tighter audits than `Target-Canary`, without destroying model utility like for `Worst-Case`.

In all experiments, we use a randomly sampled gradient from the unit ball as the canary gradient, although we did not notice any significant difference between different types of canary gradients (e.g., dirac) in preliminary experiments.

Comparing trade-off curves. We start by auditing a CNN model on CIFAR-10 for one epoch using shuffling with batch size $B = 100$, calibrating the noise multiplier to satisfy $\varepsilon = 1.0$ if Poisson subsampling was used. In Figure 4, we plot the FPR-FNR curve (95% Clopper-Pearson upper bound) from training 10^3 and 10^4 models and the curves for the corresponding theoretical (ϵ, δ)-DP at $\epsilon = 1.0$ and PLD bounds expected if Poisson subsampling had been used. We note that an *upper bound* on FPR and FNR is used to compute the corresponding *lower bound* on empirical ε_{emp} as explained previously in Section III-B. Due to computational constraints, we are only able to train 10^4 models for this experiment.

Overall, we find substantial gaps regardless of the number of observations. More precisely, even a relatively small number of observations (10^3) is enough to detect that the DP-SGD (Shuffle) implementation violates the theoretical privacy guarantees provided by the Poisson subsampling analysis. However, while auditing with the PLD curve would allow an adversary to identify that a given implementation of DP-SGD did not specifically use Poisson subsampling, it may not

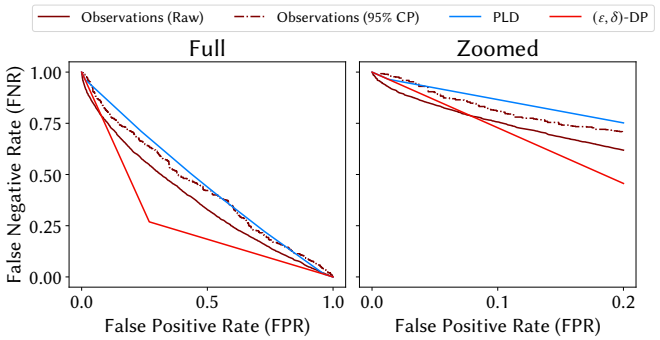

Fig. 5: Comparison of tradeoff curves for auditing DP-SGD (Shuffle) (Raw and 95% CP upper bound, 10^3 observations) vs. the theoretical tradeoff curves from (ε, δ)-DP and PLD analysis for DP-SGD (Poisson). The plot on the right is zoomed in on $0 \leq \text{FPR} \leq 0.2$.

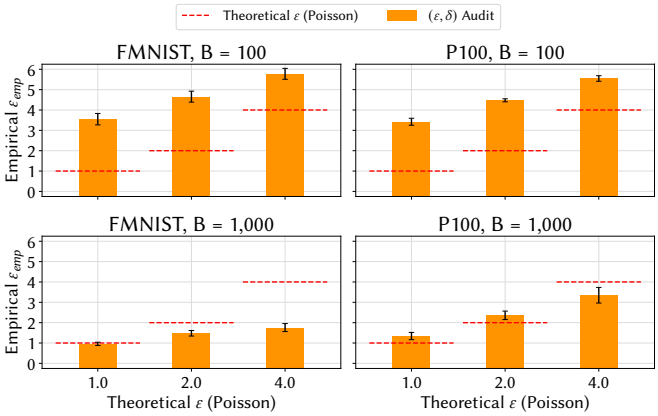

Fig. 6: Auditing DP-SGD (Shuffle) at various batch sizes (B) and privacy levels (ε).

necessarily mean that the underlying implementation does not satisfy (ϵ, δ)-DP in general.

To estimate the empirical privacy leakage observed from DP-SGD (Shuffle), we would need to audit using the (ε, δ)-DP definition instead. In Figure 4, this would correspond to comparing the observed FPR-FNR curves with the (ϵ, δ)-DP curve instead ($\epsilon = 1.0$). Here, there is a much smaller gap, with the trade-off curve for 10^4 observations only violating (ϵ, δ)-DP at FPRs < 0.05, and the trade-off curve for 10^3 not violating the theoretical DP guarantees at all. Therefore, 10^3 observations are not enough to detect a violation of (ϵ, δ)-DP guarantee: the violation can only be detected by using at least 10^4 observations at very low FPRs.

Impact of CIs. To better understand the dependence on the number of observations, we also analyze the impact of using Clopper-Pearson (CP) CIs. In Figure 5, we plot the trade-off curves from the same experiment with 10^3 observations with and without the CP upper bounds. We notice a large gap between the "raw" observed trade-off curve (w/o CP upper bounds) and the trade-off curve with CP upper bounds. Specifically, when using the trade-off curve with CP bounds, we estimate a lower bound ε_{emp}, which is always numerically smaller than the ε_{emp} estimated from the "raw" trade-off curve, thus making it more difficult to detect violations of (ε, δ)-DP. Nevertheless, in DP auditing, we are interested not only in estimating the empirical privacy leakage but also in the associated confidence level. This allows us to reduce false positives and provide assurances that the privacy violations are real, especially when debugging implementations (as we do later in Section VII). As a result, we believe it necessary to use CP intervals in our auditing experiments, even though that requires more training runs.

Varying batch size B and theoretical ε. Next, we evaluate the impact of the batch size and theoretical (Poisson) ε on the empirical privacy leakage observed from DP-SGD (Shuffle). Due to computational constraints, rather than a CNN on CIFAR-10, we audit a LeNet model on FMNIST and an MLP model on P100 over 10^6 observations. In Figure 6, we plot the empirical leakage estimates ε_{emp} observed from auditing DP-SGD (Shuffle) at various batch sizes and privacy levels ε.

With $B = 100$, the empirical privacy leakage is significantly larger than the theoretical guarantees across both datasets and privacy levels. Specifically, for $\varepsilon = 1.0, 2.0, 4.0$, privacy leakage from the models trained on FMNIST and P100 amount to, respectively, $\varepsilon_{emp} = 3.46, 4.66, 5.78$ and $3.42, 4.48, 5.55$.

However, with a larger batch $B = 1000$, we only detect small gaps in some settings. Under `Partially-Informed`, the gradients of all samples remain unchanged except for the target record and final record in each batch. Thus, this likely introduces a "bias" term when evaluating the output from each step, i.e., $o_t^i \leftarrow \langle \tilde{g}, \hat{g} \rangle$. Although Nasr et al. [42] show that this bias does not affect audits of DP-SGD (Poisson), they simply threshold the output from each step. Our auditing procedure is more complicated as we compute a likelihood ratio function across multiple steps, which we believe is more susceptible to the bias term, resulting in weaker audits than expected from BGM. However, in several settings, the empirical privacy leakage is close to or already exceeds the theoretical Poisson guarantee, suggesting that given enough observations ($\approx 10^9$), we can potentially detect larger. Due to computational constraints, we leave this to future work.

C. Varying Adversarial Capabilities

Our experiments thus far show that auditing DP-SGD (Shuffle) under `Partially-Informed` reveals large gaps between the empirical privacy leakage observed and the theoretical guarantees promised by the Poisson subsampling analysis. Next, we set out to evaluate the impact of different adversarial capabilities on the empirical privacy leakage.

In Figure 7, we report the different values of ε_{emp} when auditing DP-SGD (Shuffle) with the three different adversaries considered (see Section V). We fix the batch size to $B = 100$ and use 10^6 observations. Despite the "bias" from other training samples discussed above, the empirical privacy leakage under `Partially-Informed` is almost the same as that from the idealized `Worst-Case` threat model (which corresponds to auditing BGM) at small batch sizes. Specifically, on FMNIST, at $\varepsilon = 1.0, 2.0, 4.0$, auditing under `Partially-Informed` and `Worst-Case` yield privacy leakage estimates, respectively, of $\varepsilon_{emp} = 3.55, 4.66, 5.78$ and $\varepsilon_{emp} = 4.06, 5.18, 6.43$.

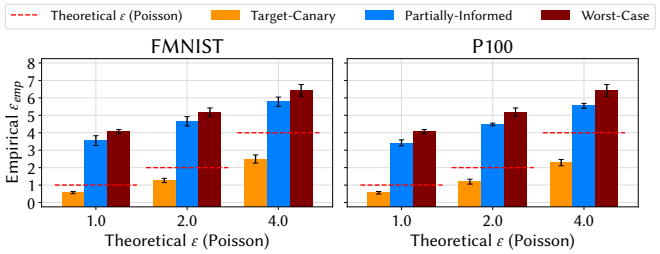

Fig. 7: Auditing DP-SGD (Shuffle) under various threat models.

However, estimates under `Target-Canary` are significantly weaker than the other two threat models. In all settings, ε_{emp} under `Target-Canary` is actually smaller than the theoretical ε guaranteed by Poisson subsampling, even though shuffling was used. This is because the negative canary gradients inserted by the `Partially-Informed` adversary are crucial in "identifying" the batch containing the target (or zero-out) record in each epoch. Specifically, in the `Partially-Informed` threat model, without any noise, the adversary aims to distinguish between $(-1, ..., +1, ..., -1)$ and $(-1, ..., 0, ..., -1)$, which has a Hamming distance of 2 with high probability (assuming the '+1' and '0' appear in different batches). Whereas under `Target-Canary`, they have to distinguish between $(0, ..., +1, ..., 0)$ and $(0, ..., 0)$, which only has a Hamming distance of 1. Informally, the distributions of outputs observed under `Target-Canary` are less distinguishable than under `Partially-Informed`, thus resulting in lower empirical privacy leakage estimates.

D. Takeaways

Our experiments confirmed that the empirical privacy leakage observed from shuffling is much larger than the theoretial privacy leakage expected from the Poisson subsampling analysis when auditing models trained using the DP-SGD (Shuffle) algorithm. Specifically, for a LeNet-5 classifier trained on FMNIST at theoretical $\varepsilon = 1.0$ in the `Partially-Informed` model, we find an empirical privacy leakage amounting to $\varepsilon_{emp} = 3.46$, almost $3.5\times$ the theoretical guarantee.

We also show that audits using a weaker adversary, i.e., `Target-Canary`, the ε_{emp} is well below the expected theoretical guarantees ε, which confirms that strong adversarial models are necessary to audit shuffling mechanisms effectively.

VII. Debugging Shuffle Implementations

Implementing DP algorithms correctly is known to be challenging [10, 33], and DP violations have been found in the wild [4, 53]. Specific to shuffling, Ponomareva et al. [47] report that for computational reasons, in many cases, datasets may not even be shuffled fully but only within a small buffer. Therefore, bugs or variations to the shuffling procedure itself may be present in DP-SGD implementations, and, naturally, this would substantially affect their privacy leakage.

Searching public GitHub repositories (see Section VII-B), we identify cases where the dataset is first batched, and the batches are then shuffled, rather than shuffling the samples before batching. As visualized in Section VII-B, this makes

Algorithm 3 Auditing Partial Shuffling (with Buffer K)

Require: Dataset, D. Number of observations, N. Target record, $(\hat{x}, \hat{y})$. Zero-out record, $(x_\perp, y_\perp)$. Significance level, α. Privacy parameter, δ.

for $i \in [\frac{N}{2}]$ **do**
  $\Theta[i] \leftarrow \text{DP-SGD}_K^{\text{PI}}(\{(\hat{x}, \hat{y})\} \cup D; -)$
  $\Theta'[i] \leftarrow \text{DP-SGD}_K^{\text{PI}}(\{(x_\perp, y_\perp)\} \cup D; -)$
end for
for $k \in \{1, 10, 20, ..., 100\}$ **do**
  for $i \in [\frac{N}{2}]$ **do**
    $[\mathbf{o}_1|...|\mathbf{o}_T] \leftarrow \Theta[i]$
    $[\mathbf{o'}_1|...|\mathbf{o'}_T] \leftarrow \Theta'[i]$
    $\mathcal{S}[i] \leftarrow \Lambda^{\text{PI}}([\mathbf{o}_1|...|\mathbf{o}_k])$
    $\mathcal{S}'[i] \leftarrow \Lambda^{\text{PI}}([\mathbf{o'}_1|...|\mathbf{o'}_k])$
  end for
  $\varepsilon_{emp}[k] \leftarrow \text{EstimateEps}(\mathcal{S}, \mathcal{S}', \alpha, \delta)$
end for
return $\max_k \varepsilon_{emp}[k]$.

Fig. 8: Comparison between a small dataset with eight records (x_1,...,x_7,x_T) being fully shuffled vs. partially shuffled with a buffer of K=4 and batch size B=2. The target record x_T is highlighted in red.

the batch with the target record substantially more identifiable in cases where the samples surrounding it are different from samples "far away" from it (in this setting, batching is done locally). While these variations to the shuffling procedure intuitively affect the privacy analysis, in this section, we verify whether our auditing procedure can identify them and determine the scale of their impact on privacy leakage.

A. Partial Shuffling

We first consider the case where datasets are only shuffled within a small buffer of K samples, as reported in [47], and report the procedure used to debug this variation in Algorithm 3. The dataset is shuffled in buffers of size K, i.e., the first K samples are shuffled and batched, followed by the next K samples, etc. Figure 8 provides a visualization of a small one-dimensional dataset that is partially shuffled.

Methodology. We assume the `Partially-Informed` adversary, where the output remains (mostly) the same at $(-1, ..., +1, ..., -1) + \mathcal{N}(0, \sigma^2\mathbb{I})$ and $(-1, ..., 0, ..., -1) + \mathcal{N}(0, \sigma^2\mathbb{I})$ for D and D', respectively. However, while in full shuffling, the '+1' and '0' can appear uniformly in any batch,

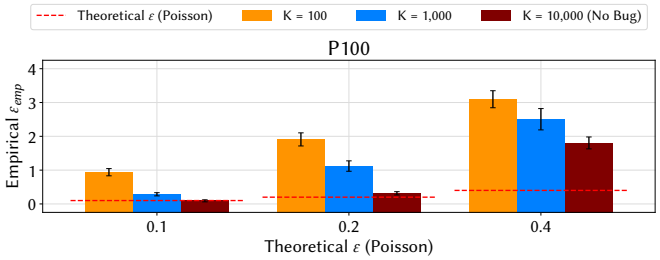

Fig. 9: Auditing DP-SGD (Shuffle) when the P100 dataset is only shuffled within a buffer of K samples. When the buffer $K = |D| = 10{,}000$, the dataset is fully shuffled.

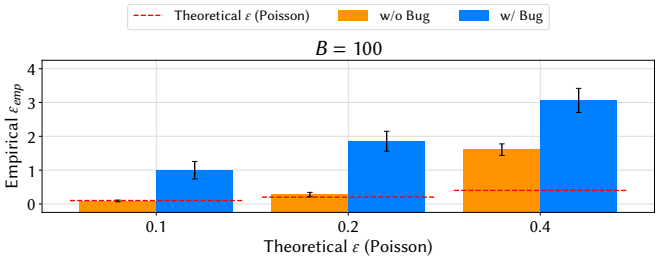

Fig. 10: Auditing DP-SGD (Shuffle) when the P100 dataset is batched before shuffling is performed.

here they can only appear in the first $\lfloor \frac{K}{B} \rfloor$ batches since the first K samples are first shuffled and then batched.

The remaining challenge in identifying this bug is that the adversary does not have access to the batch sampler $\mathcal{B}$ and, by extension, does not know K. We impose this restriction as, even in a real-world auditing setting, it may be cumbersome for an auditor to be given access to the batch sampler separately—even then, the model trainer may not faithfully use the batch sampler. Therefore, we allow the adversary to "guess" multiple buffer sizes and evaluate empirical privacy leakage from only the first K batches. The adversary then outputs the maximum empirical privacy leakage observed across all buffer sizes guessed. Theoretically, they could perform a binary search to reduce the number of guesses made; however, we assume they search over $K = \{1, 10, 20, ..., 100\}$ since the number of batches is small. Also, in theory, the empirical privacy leakage from each guess must be calculated on separate sets of observations for the 95% CI to be valid, similar to choosing the optimal threshold. However, to evaluate the maximum empirical privacy leakage achievable, we omit this step and instead report standard deviations over five runs.

We review the procedure used to debug this variation in Algorithm 3. We denote with DP-SGD$_K^{PI}$ the execution of DP-SGD with partial shuffle over a buffer of size K and with the modifications made for the `Partially-Informed` adversary as reported in Algorithm 2.

Results. In Figure 9, we report the empirical privacy leakage estimates for various buffer sizes K. We only audit the MLP model trained on P100 due to computational constraints. We experiment with small $\varepsilon = 0.1, 0.2, 0.4$ as the violations are much more significant in this regime. Specifically, at $\varepsilon = 0.1, 0.2, 0.4$, the empirical privacy leakages for full dataset shuffling $K = 10{,}000$ and partial shuffling at $K = 1{,}000$ are $\varepsilon_{emp} = 0.09, 0.31, 1.80$ and $\varepsilon_{emp} = 0.29, 1.12, 2.51$, respectively. This shows that even partial shuffling can result in substantially larger empirical privacy leakages.

However, as ε increases, the gap between full shuffling and no shuffling ($K = 100$) reduces. For instance, at $\varepsilon = 0.1, 0.2, 0.4$, the ε_{emp} values from no shuffling are $10.4\times$, $6.16\times$, and $1.72\times$ that of the ε_{emp} values from full shuffling, respectively. This indicates that for larger values of ε, the impact of not shuffling or partial shuffling steadily decreases, which in turn suggests that shuffling may not always result in strong privacy amplification.

B. Batch-then-Shuffle

Finally, we debug the variation where a dataset is first batched and then shuffled. Although this is not necessarily a known bug in the context of training private models, we find a substantial presence of this sequence of events in public repositories related to non-private ones. Specifically, we use the GitHub Code Search tool to search for occurrences of `dataset.batch(.*).shuffle(.*)`, which indicates that the dataset is first batched before it is shuffled, and compare with occurrences of `dataset.shuffle(.*).batch(.*)`.

We find approximately 10,800 files using the "correct" shuffle-then-batch implementation and 290 files (2.6%) using the batch-then-shuffle approach. As a result, we set out to preemptively explore techniques to catch these bugs.

Under the `Target-Canary` and `Partially-Informed` threat models, batch-then-shuffle is equivalent to shuffle-then-batch. This is because we expect the dot product of the privatized gradient and gradient of almost all other samples, which cannot be modified by the adversary, to be near 0 under both threat models. Thus, this bug can only be detected under the `Worst-Case` threat model. We then consider the `Worst-Case` threat model, but we let the adversary insert the gradients of the first B samples as the canary gradient $\hat{g}$ and the remaining samples as the negative canary gradient $-\hat{g}$. By doing so, the outputs from the DP-SGD algorithm are expected to be $(B, -B, ..., -B) + \mathcal{N}(0, \sigma^2\mathbb{I})$ and $(B-1, -B, ..., -B) + \mathcal{N}(0, \sigma^2\mathbb{I})$ for D and D', respectively. Note that 'B' and '$B-1$' can appear uniformly in any batch. (Again, please refer to Figure 11 for a visualization of a small one-dimensional dataset that is batched first and then shuffled.)

We adjust the likelihood ratio used under the `Worst-Case` threat model accordingly and report the audit results with/without the bug in Figure 10. Our audit can indeed easily detect this bug: at $\varepsilon = 0.1, 0.2, 0.4$, $\varepsilon_{emp} = 1.00, 1.86, 3.06$, respectively, with the bug but only $\varepsilon_{emp} = 0.09, 0.28, 1.61$, respectively, without the bug.

C. Takeaways

Our experiments show that our auditing procedure is highly extensible as it can be used to audit common variations of the shuffling procedure, more precisely, "partial shuffling" and "batch-then-shuffle." These variations remain easily detectable by our auditing method, which estimates $\varepsilon_{emp} =$

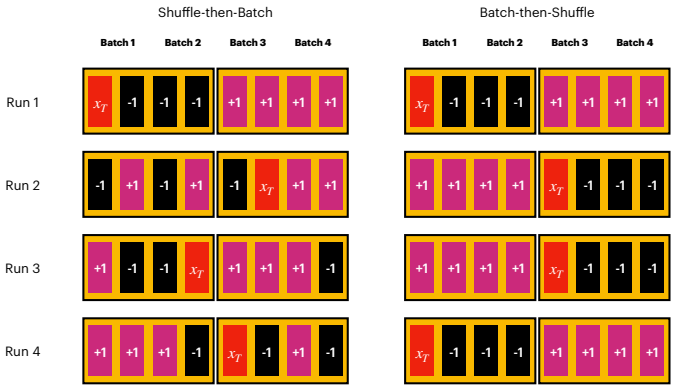

Fig. 11: Comparison between a small dataset with 8 records being shuffled then batched ("correct") vs. batched then shuffled ("wrong") with batch size $B = 4$. Target record x_T is highlighted in red.

$0.29, 1.12, 2.51$ and $\varepsilon_{emp} = 1.00, 1.86, 3.06$ for the "partial shuffling" and "batch-then-shuffle" procedure at theoretical $\varepsilon = 0.1, 0.2, 0.4$, respectively.

VIII. Related Work

To our knowledge, we are the first to audit DP-SGD with shuffling. In the following, we review relevant prior work on shuffling and auditing DP-SGD.

DP-SGD Audits. Jayaraman et al. [32] present the first DP-SGD audit using black-box inference attacks on the trained model, but only achieve loose estimates of the privacy leakage. Jagielski et al. [31] use input canaries to improve tightness and use Clopper-Pearson intervals to calculate confidence intervals for the empirical estimates. Nasr et al. [43] are the first to achieve tight DP-SGD audits using a large number of training runs, a pathological dataset, and an active white-box adversary that can insert gradient canaries at each step. To reduce the number of training runs, Zanella-Béguelin et al. [59] use credible intervals instead of Clopper-Pearson. Although [42] later questions the validity of the credible intervals, it shows that tight empirical estimates are possible even with few training runs and natural (not adversarially crafted) datasets by auditing with f-DP. Recent work on DP-SGD audits has also focused on federated learning settings [2, 40] and further reducing the number of training runs [46] to a single one [2, 41, 50] and auditing under weak threat models [3, 11].

Overall, prior work auditing DP-SGD implementations has focused on Poisson subsampling. Previous audits also typically threshold the losses [3, 31, 41–43, 46, 50] or gradients [41–43, 43, 46] directly. By contrast, we audit the shuffling setting and use likelihood ratio functions to audit DP-SGD.

Shuffling and DP-SGD. To the best of our knowledge, there is limited prior work analyzing the privacy of DP-SGD with shuffling. Chua et al. [16] analyze a simplified version of DP-SGD, i.e., the Adaptive Batch Linear Query (ABLQ) mechanism. Although they do not present a theoretical upper bound for the ABLQ mechanism with shuffling, they consider pathological settings where ABLQ could leak more privacy when using shuffling instead of Poisson subsampling. While they initially only consider a single epoch [16], in follow-up work [17], they extend their lower bound to multiple epochs. However, neither work directly analyzes DP-SGD or evaluates the gap in real-world parameter settings.

Our work takes a different approach, empirically estimating the privacy leakage from DP-SGD using DP auditing. By doing so, we derive empirical privacy leakage estimates for real-world models trained using DP-SGD (Shuffle). We also focus on the impact of real-world considerations like threat models and variations on the shuffling procedure. Overall, we believe that our work is orthogonal to [16] as our focus is on *empirical* privacy estimation, whereas theirs is on *theoretical* privacy analysis.

Shuffling. Shuffling is also used in local DP, where users randomize their inputs before sharing them for data processing. Bittau et al. [8] introduce the Encode-Shuffle-Analyze (ESA) framework, where users' randomized samples are shuffled before being processed. Cheu et al. [12] and Erlingsson et al. [26] prove that shuffling users' data improves privacy guarantees in the local DP setting. In follow-up work, Erlingsson et al. [25] apply their privacy amplification results to the ESA framework, while Balle et al. [6] improve on and generalize results from [26]. Finally, Feldman et al. [27, 28] and Wang et al. [54] present a nearly optimal analysis of shuffling in the local DP setting. Overall, the privacy analysis for (ε, δ)-DP mechanisms typically used in central DP is far less understood than that of shuffling for local DP mechanisms, which typically guarantee pure DP (i.e., $\delta = 0$).

IX. Discussion and Conclusion

This paper presented the first audit of DP-SGD implementations that shuffle the training data and deterministically iterate over fixed-size batches, which we denoted as DP-SGD (Shuffle). Arguably, this addressed an important research problem as using shuffling to sample batches in DP-SGD has become common [19, 39, 47], mostly due to better computational efficiency, even though theoretical guarantees are reported as if Poisson subsampling was used, which we referred to as DP-SGD (Poisson). This makes it crucial to investigate the gap between the empirical privacy leakage from the former and the theoretical guarantees of the latter.

We introduced new auditing techniques and audited DP-SGD (Shuffle) with different parameter settings and threat models. More precisely, we experimented with 101 settings from SOTA models using shuffling [19, 39], finding gaps in two-thirds of them—in some cases, the empirical privacy leakage from DP-SGD (Shuffle) was up to 4 times higher than the theoretical guarantee from DP-SGD (Poisson). We also showed how these discrepancies can depend on the batch size and the strength of the adversary, with smaller batch sizes and stronger threat models yielding larger gaps. Finally, we used our techniques to detect the presence of bugs in the implementations of the shuffling procedure. We investigated two common bugs [47] found in public code repositories of non-private model training, showing that our auditing framework effectively identifies them and detects even higher empirical privacy leakages in their presence.

Implications and Recommendations for Practitioners. The measurable differences between the empirical and the theoretical guarantees shows that prior work [19, 39] substantially overestimated the privacy guarantees of some SOTA private models. Consequently, our work calls into question the validity of reporting theoretical guarantees using DP-SGD (Poisson) while implementing algorithms that use DP-SGD (Shuffle), reiterating the importance of formally analyzing the privacy guarantees provided by shuffling the dataset. Concretely, we recommend practitioners avoid shuffling until tighter guarantees are proven. Recently proposed alternative sampling schemes like Balls-in-Bins [13, 15] could also be used, as they could balance shuffling's computational efficiency with provably tight guarantees.

Adversarial Models. Our audits use adversaries in active white-box models; specifically, we assume they can insert a target sample of their choice and view the final trained model, and can insert (varying sizes of) canary gradients at each step and observe all intermediate models. While we introduce an adversarial model unique to shuffling, i.e., `Partially-Informed`, this serves as a hypothetical middle-ground between the `Worst-Case` adversary and a weaker one (`Target-Canary`), and may not necessarily reflect an actual adversary. Nevertheless, as discussed in Section V-A, this primarily serves as a useful auditing tool, enabling us to identify the impact that the bias from other (non-target) samples has on our novel likelihood-based auditing method.

Overall, although not all the adversaries we consider may always have the corresponding capabilities against models deployed in production, recall that provably correct DP guarantees are meant to be robust in *worst-case* settings, making our adversaries valid in the context of auditing. In fact, similar assumptions were made in prior DP-SGD audits [4, 42, 43].

Limitations & Future Work. One main limitation of our work is the computational cost required to run the audit. Specifically, since shuffling does not provide any improved f-DP or μ-GDP guarantees, we require training runs in the order of thousands and millions. Recent work presented audits of DP-SGD (Poisson) with just one run [2, 41, 46, 50]; however, these techniques may not directly apply to DP-SGD (Shuffle) and underestimate privacy leakage even under powerful threat models. Thus, it is not yet clear how to adapt them to our setting. In follow-up work, we plan to investigate how to minimize the number of training runs required to audit DP-SGD (Shuffle) accurately, which would also enable us to audit larger deep learning models.

In addition, our audits are weaker for larger batch sizes, which we believe is mainly due to the bias introduced by the "other" samples. We plan to reduce this bias term using debiasing techniques (e.g., [51]) to make our audits effective even for large batch sizes.

Finally, due to computational constraints, we only audited one epoch of DP-SGD (Shuffle) and therefore did not measure model utility. In the future, we plan to audit state-of-the-art models built using DP-SGD (Shuffle) directly, which will also allow us to measure drops in utility with different adversaries. As mentioned, we expect utility to plummet with `Worst-Case` but remain stable with `Partially-Informed` and `Target-Canary` adversaries.

Acknowledgements. This work has been supported by the National Science Scholarship (PhD) from the Agency for Science Technology and Research, Singapore.

REFERENCES

[1] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep Learning with Differential Privacy. In *CCS*, 2016.

[2] G. Andrew, P. Kairouz, S. Oh, A. Oprea, H. B. McMahan, and V. Suriyakumar. One-shot Empirical Privacy Estimation for Federated Learning. In *ICLR*, 2024.

[3] M. S. M. S. Annamalai and E. De Cristofaro. Nearly Tight Black-Box Auditing of Differentially Private Machine Learning. In *NeurIPS*, 2024.

[4] M. S. M. S. Annamalai, G. Ganev, and E. De Cristofaro. "What do you want from theory alone?" Experimenting with Tight Auditing of Differentially Private Synthetic Data Generation. In *USENIX Security*, 2024.

[5] B. Balle, G. Barthe, and M. Gaboardi. Privacy Amplification by Subsampling: Tight Analyses via Couplings and Divergences. In *NeurIPS*, 2018.

[6] B. Balle, J. Bell, A. Gascón, and K. Nissim. The Privacy Blanket of the Shuffle Model. In *CRYPTO*, 2019.

[7] B. Balle, L. Berrada, S. De, S. Ghalebikesabi, J. Hayes, A. Pappu, S. L. Smith, and R. Stanforth. JAX-Privacy: Algorithms for Privacy-Preserving Machine Learning in JAX. https://github.com/google-deepmind/jax_privacy, 2022.

[8] A. Bittau, Ú. Erlingsson, P. Maniatis, I. Mironov, A. Raghunathan, D. Lie, M. Rudominer, U. Kode, J. Tinnes, and B. Seefeld. Prochlo: Strong Privacy for Analytics in the Crowd. In *SOPS*, 2017.

[9] N. Carlini, S. Chien, M. Nasr, S. Song, A. Terzis, and F. Tramèr. Membership Inference Attacks From First Principles. In *IEEE S&P*, 2022.

[10] T. Cebere. Privacy Leakage at low sample size. https://github.com/pytorch/opacus/issues/571, 2023.

[11] T. Cebere, A. Bellet, and N. Papernot. Tighter Privacy Auditing of DP-SGD in the Hidden State Threat Model. *arXiv:2405.14457*, 2024.

[12] A. Cheu, A. Smith, J. Ullman, D. Zeber, and M. Zhilyaev. Distributed Differential Privacy via Shuffling. In *Eurocrypt*, 2019.

[13] C. A. Choquette-Choo, A. Ganesh, S. Haque, T. Steinke, and A. G. Thakurta. Near-Exact Privacy Amplification for Matrix Mechanisms. In *ICLR*, 2025.

[14] C. A. Choquette-Choo, A. Ganesh, R. McKenna, H. B. McMahan, J. Rush, A. Guha Thakurta, and Z. Xu. (Amplified) Banded Matrix Factorization: A unified approach to private training. In *NeurIPS*, 2024.

[15] L. Chua, B. Ghazi, C. Harrison, E. Leeman, P. Kamath, R. Kumar, P. Manurangsi, A. Sinha, and C. Zhang. Balls-and-bins sampling for DP-SGD. In *AISTATS*, 2025.

[16] L. Chua, B. Ghazi, P. Kamath, R. Kumar, P. Manurangsi, A. Sinha, and C. Zhang. How Private are DP-SGD Implementations? In *ICML*, 2024.

[17] L. Chua, B. Ghazi, P. Kamath, R. Kumar, P. Manurangsi, A. Sinha, and C. Zhang. Scalable DP-SGD: Shuffling vs. Poisson Subsampling. *arXiv:2411.04205*, 2024.

[18] C. J. Clopper and E. S. Pearson. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, 1934.

[19] S. De, L. Berrada, J. Hayes, S. L. Smith, and B. Balle. Unlocking High-Accuracy Differentially Private Image Classification through Scale. *arXiv:2204.13650*, 2022.
[20] E. Debenedetti, G. Severi, N. Carlini, C. A. Choquette-Choo, M. Jagielski, M. Nasr, E. Wallace, and F. Tramèr. Privacy Side Channels in Machine Learning Systems. In *USENIX*, 2024.
[21] Z. Ding, Y. Wang, G. Wang, D. Zhang, and D. Kifer. Detecting Violations of Differential Privacy. In *CCS*, 2018.
[22] J. Dong, A. Roth, and W. J. Su. Gaussian Differential Privacy. *arXiv:1905.02383*, 2019.
[23] F. Dörmann, O. Frisk, L. N. Andersen, and C. F. Pedersen. Not All Noise is Accounted Equally: How Differentially Private Learning Benefits from Large Sampling Rates. In *MLSP*, 2021.
[24] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating Noise to Sensitivity in Private Data Analysis. In *TCC*, 2006.
[25] Ú. Erlingsson, V. Feldman, I. Mironov, A. Raghunathan, S. Song, K. Talwar, and A. Thakurta. Encode, Shuffle, Analyze Privacy Revisited: Formalizations and Empirical Evaluation. *arXiv:2001.03618*, 2020.
[26] Ú. Erlingsson, V. Feldman, I. Mironov, A. Raghunathan, K. Talwar, and A. Thakurta. Amplification by Shuffling: From Local to Central Differential Privacy via Anonymity. In *SODA*, 2019.
[27] V. Feldman, A. McMillan, and K. Talwar. Hiding Among the Clones: A Simple and Nearly Optimal Analysis of Privacy Amplification by Shuffling. In *FOCS*, 2022.
[28] V. Feldman, A. McMillan, and K. Talwar. Stronger Privacy Amplification by Shuffling for Renyi and Approximate Differential Privacy. In *SODA*, 2023.
[29] V. Feldman and M. Shenfeld. Privacy amplification by random allocation. *arXiv:2502.08202*, 2025.
[30] Google. TensorFlow Privacy. https://github.com/tensorflow/privacy, 2019.
[31] M. Jagielski, J. Ullman, and A. Oprea. Auditing Differentially Private Machine Learning: How Private is Private SGD? In *NeurIPS*, 2020.
[32] B. Jayaraman and D. Evans. Evaluating differentially private machine learning in practice. In *USENIX Security*, 2019.
[33] M. Johnson. Fix prng key reuse in differential privacy example. https://github.com/google/jax/pull/3646, 2020.
[34] P. Kairouz, B. McMahan, S. Song, O. Thakkar, A. Thakurta, and Z. Xu. Practical and Private (Deep) Learning Without Sampling or Shuffling. In *ICML*, 2021.
[35] P. Kairouz, S. Oh, and P. Viswanath. The Composition Theorem for Differential Privacy. In *ICML*, 2015.
[36] A. Koskela, J. Jälkö, and A. Honkela. Computing Tight Differential Privacy Guarantees Using FFT. In *AISTATS*, 2020.
[37] A. Krizhevsky. Learning Multiple Layers of Features from Tiny Images. https://www.cs.utoronto.ca/~kriz/learning-features-2009-TR.pdf, 2009.
[38] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradientbased learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 1998.
[39] X. Li, F. Tramèr, P. Liang, and T. Hashimoto. Large Language Models Can Be Strong Differentially Private Learners. In *ICLR*, 2022.
[40] S. Maddock, A. Sablayrolles, and P. Stock. CANIFE: Crafting Canaries for Empirical Privacy Measurement in Federated Learning. In *ICLR*, 2023.
[41] S. Mahloujifar, L. Melis, and K. Chaudhuri. Auditing f-Differential Privacy in One Run. In *ICML*, 2025.
[42] M. Nasr, J. Hayes, T. Steinke, B. Balle, F. Tramèr, M. Jagielski, N. Carlini, and A. Terzis. Tight Auditing of Differentially Private Machine Learning. In *USENIX Security*, 2023.
[43] M. Nasr, S. Songi, A. Thakurta, N. Papernot, and N. Carlini. Adversary Instantiation: Lower Bounds for Differentially Private Machine Learning. In *IEEE S&P*, 2021.
[44] J. Neyman and E. S. 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.
[45] OpenXLA. XLA (Accelerated Linear Algebra). https://github.com/openxla/xla, 2024.
[46] K. Pillutla, G. Andrew, P. Kairouz, H. B. McMahan, A. Oprea, and S. Oh. Unleashing the Power of Randomization in Auditing Differentially Private ML. In *NeurIPS*, 2024.
[47] N. Ponomareva, S. Vassilvitskii, Z. Xu, B. McMahan, A. Kurakin, and C. Zhang. How to DP-fy ML: A Practical Tutorial to Machine Learning with Differential Privacy. In *KDD*, 2023.
[48] A. S. Shamsabadi, G. Tan, T. I. Cebere, A. Bellet, H. Haddadi, N. Papernot, X. Wang, and A. Weller. Confidential-DPproof: Confidential Proof of Differentially Private Training. In *ICLR*, 2024.
[49] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership Inference Attacks against Machine Learning Models. In *IEEE S&P*, 2017.
[50] T. Steinke, M. Nasr, and M. Jagielski. Privacy Auditing with One (1) Training Run. In *NeurIPS*, 2024.
[51] Y. Tong, J. Ye, S. Zarifzadeh, and R. Shokri. How much of my dataset did you use? Quantitative Data Usage Inference in Machine Learning. In *ICLR*, 2025.
[52] F. Tramèr and D. Boneh. Differentially Private Learning Needs Better Features (or Much More Data). In *ICLR*, 2021.
[53] F. Tramèr, A. Terzis, T. Steinke, S. Song, M. Jagielski, and N. Carlini. Debugging Differential Privacy: A Case Study for Privacy Auditing. *arXiv:2202.12219*, 2022.
[54] C. Wang, B. Su, J. Ye, R. Shokri, and W. Su. Unified Enhancement of Privacy Bounds for Mixture Mechanisms via f-Differential Privacy. In *NeurIPS*, 2023.
[55] Z. Xiang, T. Wang, and D. Wang. Privacy Audit as Bits Transmission:(Im) possibilities for Audit by One Run. In *USENIX Security*, 2025.
[56] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv:1708.07747*, 2017.
[57] J. Ye, A. Maddi, S. K. Murakonda, V. Bindschaedler, and R. Shokri. Enhanced Membership Inference Attacks against Machine Learning Models. In *CCS*, 2022.
[58] A. Yousefpour, I. Shilov, A. Sablayrolles, D. Testuggine, K. Prasad, M. Malek, J. Nguyen, S. Ghosh, A. Bharadwaj, J. Zhao, et al. Opacus: User-friendly Differential Privacy Library in PyTorch. *arXiv:2109.12298*, 2021.
[59] S. Zanella-Béguelin, L. Wutschitz, S. Tople, A. Salem, V. Rühle, A. Paverd, M. Naseri, B. Köpf, and D. Jones. Bayesian Estimation of Differential Privacy. In *ICML*, 2023.
[60] Y. Zhu, J. Dong, and Y.-X. Wang. Optimal Accounting of Differential Privacy via Characteristic Function. In *AISTATS*, 2022.

APPENDIX A
DIFFERENTIALLY PRIVATE STOCHASTIC GRADIENT DESCENT (DP-SGD)

In Algorithm 4, we review DP-SGD [1]'s pseudo-code. Then, in Algorithm 5, we report that of Batched Gaussian Mechanism (BGM), a heavily simplified version of DP-SGD adapted from [16] to develop principled tight auditing techniques for shuffling under an idealized setting.

Algorithm 4 Differentially Private Stochastic Gradient Descent (DP-SGD) [1]

Require: Dataset, D. Epochs, E. Batch Size, B. Learning rate, η. Batch sampler, $\mathcal{B}$. Loss function, ℓ. Initial model parameters, θ_0. Noise multiplier, σ. Clipping norm, C.

$T \leftarrow |D|/B$
for $i \in [E]$ **do**
  $\theta_1^i \leftarrow \theta_T^{i-1}$
  Sample batches $B_1, ..., B_T \leftarrow \mathcal{B}(D, B)$
  for $t \in [T]$ **do**
    for $(x_j, y_j) \in B_t$ **do**
      $g_j \leftarrow \nabla\ell((x_j, y_j); \theta_t^i)$
      $\bar{g}_j \leftarrow g_j / \max(1, \frac{||g_j||_2}{C})$
    end for
    $\tilde{g} \leftarrow \frac{1}{B}\left(\sum_j \bar{g}_j + \mathcal{N}(0, C^2\sigma^2\mathbb{I})\right)$
    $\theta_{t+1}^i \leftarrow \theta_t^i - \eta\tilde{g}$
  end for
end for
return θ_T^E

Algorithm 5 Batched Gaussian Mechanism (BGM)

Require: Dataset, $D = (x_1, ..., x_N) \in [-1, +1]^N$. Batch Size, B. Number of epochs, E. Batch sampler, $\mathcal{B}$. Noise multiplier, σ.

$T \leftarrow |D|/B$
for $i \in [E]$ **do**
  Sample batches $B_1, ..., B_T \leftarrow \mathcal{B}(D, B)$
  for $t \in [T]$ **do**
    $\tilde{g}_t^i \leftarrow \sum_{x_i \in B_t} x_i + \mathcal{N}(0, \sigma^2)$
  end for
end for
return $\begin{bmatrix} \tilde{g}_1^1 & \dots & \tilde{g}_T^1 \\ \vdots & \ddots & \vdots \\ \tilde{g}_1^E & \dots & \tilde{g}_T^E \end{bmatrix}$

APPENDIX B
MODEL ARCHITECTURES

In our experiments, we audit several (shallow) Convolutional Neural Networks (CNNs) and an MLP model corresponding to different datasets. More precisely, for the CIFAR-10 dataset, we use the CNN from Dörmann et al. [23], who make minor modifications to the CNNs previously used by Tramèr and Boneh [52]. For the FMNIST dataset, we use a small LeNet-5 model [38]. Finally, for the P100 dataset, we use an MLP model with 32 hidden neurons and ReLU activations.

Exact model architectures for CIFAR-10 and FMNIST are reported in Tables II and III, respectively.

APPENDIX C
WORST-CASE VS PATHOLOGICAL DATASET

We now briefly explain how our Worst-Case adversary can be modeled with the Pathological Dataset adversary used in prior work [43]. The Pathological Dataset adversary crafts the malicious dataset by making sure the dataset (not including the target sample) is labeled perfectly by the initial model. This ensures that the gradients of all other samples are zero, except for the target sample. Furthermore, to ensure that subsequent model updates do not disrupt the gradients, the Pathological Dataset also sets the learning rate to 0.

On the other hand, in our work, the Worst-Case adversary crafts malicious gradients for all samples that are equal in magnitude but opposite in direction to the target sample. This can be modeled by the Pathological Dataset adversary by first calculating the gradient of a given target sample with respect to the initial model parameters. Then, the other samples can be crafted by optimizing an input sample with respect to the target sample's gradient in the opposite direction and repeating them N times. Finally, the learning rate can be set to 0 to ensure that future model updates do not disrupt the gradients. Therefore, we can ensure that the gradient of the target sample is always equal in magnitude and opposite in direction to the target sample throughout the training process.

Layer	Parameters
Convolution	32 filters of 3x3, stride 1, padding 1
Convolution	32 filters of 3x3, stride 1, padding 1
Max-Pooling	2x2, stride 2, padding 0
Convolution	64 filters of 3x3, stride 1, padding 1
Convolution	64 filters of 3x3, stride 1, padding 1
Max-Pooling	2x2, stride 2, padding 0
Convolution	128 filters of 3x3, stride 1, padding 1
Convolution	128 filters of 3x3, stride 1, padding 1
Max-Pooling	2x2, stride 2, padding 0
Fully connected	128 units
Fully connected	2 units

TABLE II: Shallow CNN model for CIFAR-10 with Tanh activations.

Layer	Parameters
Convolution	6 filters of 5x5, stride 1, padding 2
Avg-Pooling	2x2
Convolution	16 filters of 5x5, stride 1
Avg-Pooling	2x2
Fully connected	120 units
Fully connected	84 units
Fully connected	2 units

TABLE III: LeNet-5 model for FMNIST with Tanh activations.