Practical and Private (Deep) Learning without Sampling or Shuffling

Peter Kairouz* Brendan McMahan* Shuang Song* Om Thakkar*
Abhradeep Thakurta* Zheng Xu*

March 2, 2021

Abstract

We consider training models with differential privacy (DP) using mini-batch gradients. The existing state-of-the-art, Differentially Private Stochastic Gradient Descent (DP-SGD), requires *privacy amplification by sampling or shuffling* to obtain the best privacy/accuracy/computation trade-offs. Unfortunately, the precise requirements on exact sampling and shuffling can be hard to obtain in important practical scenarios, particularly federated learning (FL). We design and analyze a DP variant of Follow-The-Regularized-Leader (DP-FTRL) that compares favorably (both theoretically and empirically) to amplified DP-SGD, while allowing for much more flexible data access patterns. DP-FTRL does not use any form of privacy amplification.

1 Introduction

Differentially private stochastic gradient descent (DP-SGD) [1, 6, 65] has become state-of-the-art in training private (deep) learning models [1, 25, 27, 50, 55, 68]. It operates by running stochastic gradient descent [59] on noisy mini-batch gradients[1], with the noise calibrated such that it ensures differential privacy. The privacy analysis heavily uses tools like *privacy amplification by sampling/shuffling* [1, 6, 24, 30, 40, 69, 73] to obtain the best privacy/utility trade-offs. Such amplification tools require that each mini-batch is a perfectly (uniformly) random subset of the training data. This assumption can make practical deployment prohibitively hard, especially in the context of distributed settings like federated learning (FL) where one has little control on which subset of the training data one sees at any time [5, 38].

We propose a new online learning [32, 62] based DP algorithm, *differentially private follow-the-regularized-leader* (DP-FTRL), that has privacy/utility/computation trade-offs that are competitive with DP-SGD, and does not rely on privacy amplification. DP-FTRL *significantly outperforms* un-amplified DP-SGD at all privacy levels. In the higher-accuracy / lower-privacy regime, DP-FTRL outperforms even *amplified* DP-SGD. We emphasize that in the context of ML applications, using a DP mechanism even with a large ε is practically much better for privacy than using a non-DP mechanism [35, 52, 64, 67].

Privacy amplification and its perils: At a high-level, DP-SGD can be thought of as an iterative noisy state update procedure for T steps operating over mini-batches of the training data. For a time step $t \in [T]$ and an arbitrary mini-batch of size k from a data set D of size n, let σ_t be the standard deviation of the noise needed in the t^{th} update to satisfy ε_t-differential privacy. If the mini-batch is chosen *u.a.r. and i.i.d.* from D at each time step[2] t, then privacy amplification by sampling [1, 6, 40, 69] allows one to scale down the noise to $\sigma_t \cdot (k/n)$, while still ensuring ε_t-differential privacy.[3] Such amplification is crucial for DP-SGD to obtain state-of-the-art models in practice [1, 55, 68] when $k \ll n$.

*Google. {kairouz, mcmahan, shuangsong, omthkkr, athakurta, xuzheng}@google.com

[1] Gradient computed on a subset of the training examples, also called a mini-batch.

[2] One can also create a mini-batch with Poisson sampling [1, 49, 73], except the batch size is now a random variable. For brevity, we focus on the fixed batch setting.

[3] A similar argument holds for amplification by shuffling [24, 30], when the data are uniformly shuffled at the beginning of every epoch.We do not consider privacy amplification by iteration [28] in this paper, as it only applies to smooth convex functions.

There are two major bottlenecks for such deployments: i) For large data sets, achieving uniform sampling/shuffling of the mini-batches in every round (or epoch) can be prohibitively expensive in terms of computation and/or engineering complexity, ii) In distributed settings like federated learning (FL) [45], uniform sampling/shuffling may be infeasible to achieve because of widely varying available population at each time step. Our work answers the following question in affirmative: *Can we design an algorithm that does not rely on privacy amplification, and hence allows data to be accessed in an arbitrary order, while providing privacy/utility/computation trade-offs competitive with DP-SGD?*

DP-FTRL and amplification-free model training: DP-FTRL can be viewed as a differentially private variant of the follow-the-regularized-leader (FTRL) algorithm [17, 44, 72]. The main idea in DP-FTRL is to use the *tree aggregation trick* [14, 23] to add noise to the sum of mini-batch gradients, in order to ensure privacy. Crucially, it deviates from DP-SGD by adding correlated noise across time steps, as opposed to independent noise. This particular aspect of DP-FTRL allows it to get strong privacy/utility trade-off without relying on privacy amplification.

Federated Learning (FL) and DP-FTRL: There has been prior work [5, 57] detailing challenges for obtaining strong privacy guarantees that incorporate limited availability of participating clients in real-world applications of Federated Learning. Although there exist techniques like the Random Check-Ins [5] that obtain privacy amplification for FL settings, implementing such techniques may still require clients to keep track of the number of training rounds being completed at the server during their period(s) of availability to be able to uniformly randomize their participation. On the other hand, since the privacy guarantees of DP-FTRL (Algorithm 1) do not depend on any type of privacy amplification, it does not require any local/central randomness apart from noise addition to the model updates.

Appendices A and Section 2 describe additional related work and background, respectively.

1.1 Problem Formulation

Suppose we have a stream of data samples $D = [d_1, \ldots, d_n] \in \mathcal{D}^n$, where $\mathcal{D}$ is the domain of data samples, and a loss function $\ell : \mathcal{C} \times \mathcal{D} \to \mathbb{R}$, where $\mathcal{C} \in \mathbb{R}^p$ is the space of all models. We consider the following two problem settings.

Regret Minimization: At every time step $t \in [n]$, while observing samples $[d_1, \ldots, d_{t-1}]$, the algorithm $\mathcal{A}$ outputs a model $\theta_t \in \mathcal{C}$ which is used to predict on example d_t. The performance of $\mathcal{A}$ is measured in terms of regret against an arbitrary post-hoc comparator $\theta^* \in \mathcal{C}$:

$$R_D(\mathcal{A}; \theta^*) = \frac{1}{n} \sum_{t=1}^{n} \ell(\theta_t; d_t) - \frac{1}{n} \sum_{t=1}^{n} \ell(\theta^*; d_t). \tag{1}$$

We consider the algorithm $\mathcal{A}$ low-regret if $R_D(\mathcal{A}; \theta^*) = o(1)$. To ensure a low-regret algorithm, we will assume $\|\nabla \ell(\theta; d)\|_2 \leq L$ for any data sample d, and any models $\theta \in \mathcal{C}$. We consider both *adversarial regret*, where the data sample d_t are drawn adversarially based on the past output $\{\theta_1, \ldots, \theta_t\}$ [32], and *stochastic regret* [33], where the data samples in D are drawn i.i.d. from some fixed distribution τ.

Excess Risk Minimization: In this setting, we look at the problem of minimizing the excess population risk. Assuming the data set D is sampled i.i.d. from a distribution τ, and the algorithm $\mathcal{A}$ outputs $\widehat{\theta} \in \mathcal{C}$, we want to minimize

$$\mathsf{PopRisk}(\mathcal{A}) = \mathbb{E}_{d \sim \tau} \ell(\widehat{\theta}; d) - \min_{\theta \in \mathcal{C}} \mathbb{E}_{d \sim \tau} \ell(\theta; d). \tag{2}$$

All the algorithms in this paper guarantee differential privacy [21, 22] and Rényi differential privacy [51] (See Section 2 for details). The definition of a single data record can be one training example (a.k.a., *example level* privacy), or a group of training examples from one individual (a.k.a., *user level* privacy). Except for the empirical evaluations in the FL setting, we focus on example level privacy.

Definition 1.1 (Differential privacy [21, 22])**.** *A randomized algorithm $\mathcal{A}$ is (ε, δ)-differentially private if for any neighboring data sets D, D' that differ in one record, and for any event $\mathcal{S}$ in the output range of $\mathcal{A}$, we have*

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

where the probability is over the randomness of $\mathcal{A}$.

Table 1: Best known regret guarantees. Here, the high probability means w.p. at least $1-\beta$ over the the randomness of the algorithm. The expected regret is an expectation over the random choice of the data set and the randomness of the algorithm.

Class	Adversarial Regret		Stochastic Regret	
	Expected	High probability	Expected	High probability
Least-squares (and linear)	$O\left(\left(\frac{1}{\sqrt{n}}+\frac{\sqrt{p}}{\varepsilon n}\right)\cdot \mathsf{polylog}\left(\frac{1}{\delta},n\right)\right)$ [3]	Same as general convex	$O\left(\left(\frac{1}{\sqrt{n}}+\frac{\sqrt{p}}{\varepsilon n}\right)\cdot \mathsf{polylog}\left(\frac{1}{\delta},n\right)\right)$ [3]	$O\left(\left(\frac{1}{\sqrt{n}}+\frac{\sqrt{p}}{\varepsilon n}\right)\cdot \mathsf{polylog}\left(\frac{1}{\delta},n,\frac{1}{\beta}\right)\right)$ [Theorem 4.3]
General convex	Constrained and unconstrained: $O\left(\left(\frac{1}{\sqrt{n}}+\frac{p^{1/4}}{\sqrt{\varepsilon n}}\right)\cdot \mathsf{polylog}\left(\frac{1}{\delta},n,\frac{1}{\beta}\right)\right)$ [Theorem 4.1]			

1.2 Our Contributions

Our primary contribution in this paper is a private online learning algorithm: differentially private follow-the-regularized leader (DP-FTRL) (Algorithm 1). We provide tighter privacy/utility trade-offs based on DP-FTRL (see Table 1 for a summary), and show how it can be easily adapted to train (federated) deep learning models, with comparable, and sometimes even better privacy/utility/computation trade-offs as DP-SGD. We summarize these contributions below.

DP-FTRL algorithm: We provide DP-FTRL, a differentially private variant of the Follow-the-regularized-leader (FTRL) algorithm [32, 44, 47, 62] for online convex optimization (OCO). We also provide a variant called the momentum DP-FTRL that has superior performance in practice. [3] provided a instantiation of DP-FTRL specific to linear losses. [63] provided an algorithm similar to DP-FTRL, where instead of just linearizing the loss, a quadratic approximation to the regularized loss was used.

Regret guarantees: In the adversarial OCO setting (Section 4.1), compared to prior work [3, 37, 63], DP-FTRL has the following major advantages. First, it improves the best known regret guarantee in [63] by a factor of $\sqrt{\varepsilon}$ (from $\widetilde{O}\left(\sqrt{\frac{\sqrt{p}}{\varepsilon^2 n}}\right)$ to $\widetilde{O}\left(\sqrt{\frac{\sqrt{p}}{\varepsilon n}}\right)$, when $\varepsilon \leq 1$). This improvement is significant because it *distinguishes centrally private OCO from locally private* [26, 40, 70] OCO[4]. Second, unlike [63], DP-FTRL (and its analysis) extends to the unconstrained setting $\mathcal{C}=\mathbb{R}^p$. Also, in the case of composite losses [18, 44, 46, 72], i.e., where the loss functions are of the form $\ell(\theta;d_t)+r_t(\theta)$ with $r:\mathcal{C}\to\mathbb{R}^+$ (e.g., $\|\cdot\|_1$) being a convex regularizer, DP-FTRL has a regret guarantee for the losses $\ell(\theta;d_t)$'s of form: (regret bound without the r_t's) $+\frac{1}{n}\sum_{t=1}^{n} r_t(\theta^*)$.

In the stochastic OCO setting (Section 4.2), we show that for least-square losses (where $\ell(\theta;d_t)=(y_t-\langle \mathbf{x}_t,\theta\rangle)^2$ with $d_t=(\mathbf{x}_t,y_t)$) and linear losses (when $\ell(\theta;d_t)=\langle d_t,\theta\rangle$), a variant of DP-FTRL achieves regret of the form $O\left(\left(\frac{1}{\sqrt{n}}+\frac{\sqrt{p}}{\varepsilon n}\right)\cdot \mathsf{polylog}\left(\frac{1}{\delta},n,\frac{1}{\beta}\right)\right)$ with probability $1-\beta$ over the randomness of algorithm. Our guarantees are strictly high-probability guarantees, i.e., the regret only depends on $\mathsf{polylog}(1/\beta)$.

Population risk guarantees: In Section 4.3, using the standard online-to-batch conversion [13, 61], we obtain a population risk guarantee for DP-FTRL. For general Lipschitz convex losses, the population risk for DP-FTRL in Theorem C.5 is same as that in [6, Appendix F] (up to logarithmic factors), but the advantage of DP-FTRL is that it is a single pass algorithm (over the data set D), as opposed to requiring n passes over the data. *Thus, we provide the best known population risk guarantee for a single pass algorithm.* While the results in [7, 9, 29] have a tighter (and optimal) excess population risk of $\widetilde{\Theta}(1/\sqrt{n}+\sqrt{p}/(\varepsilon n))$, they either require smoothness property of the convex function for a single pass algorithm, or need to make n-passes over the data. For restricted classes like linear and least-squared losses, DP-FTRL can achieve the optimal population risk via the tighter stochastic regret guarantee. Whether DP-FTRL can achieve the optimal excess population risk in the general convex setting is left as an open problem.

[4] Although not stated formally in the literature, a simple argument shows that locally private SGD [19] can achieve the same regret as in [63].

Empirical contributions: In Section 5, we study some trade-offs between privacy/utility/computation for DP-FTRL and DP-SGD. We conduct our experiments on four benchmark data sets: MNIST, CIFAR-10, EMNIST, and StackOverflow. We start by fixing the computation available to the techniques, and observing privacy/utility trade-offs. We find that DP-FTRL achieves better utility compared to DP-SGD for moderate to large ε. In scenarios where amplification cannot be ensured (e.g., due to practical/implementation constraints), DP-FTRL provides substantially better performance as compared to unamplified DP-SGD. Moreover, we show that with a modest increase in the computation cost, DP-FTRL, without any need for amplification, can match the performance of amplified DP-SGD. Next, we focus on privacy/computation trade-offs for both the techniques when a utility target is desired. We show that DP-FTRL can provide better trade-offs compared to DP-SGD for various accuracy targets, which can result in significant savings in privacy/computation cost as the size of data sets becomes limited.

To shed light on the empirical efficacy of DP-FTRL (in comparison) to DP-SGD, in Section 3.2, we show that a variant of DP-SGD (with correlated noise) can be viewed as an equivalent formulation of DP-FTRL in the unconstrained setting ($\mathcal{C} = \mathbb{R}^p$). In the case of traditional DP-SGD [6], the scale of the noise added per-step $t \in [n]$ is asymptotically same as that of DP-FTRL once $t = \omega(n)$.

2 Background

Differential Privacy: Throughout the paper, we use the notion of approximate differential privacy [21, 22] and Rényi differential privacy (RDP) [1, 51]. For meaningful privacy guarantees, ε is assumed to be a small constant, and $\delta \ll 1/|D|$.

Definition 2.1 (RDP [1, 51])**.** *A randomized algorithm $\mathcal{A}$ is (α, ε)-RDP if for any pair of neighboring datasets D, D' that differ in one record, we have*

$$\frac{1}{\alpha - 1} \log \mathop{\mathbf{E}}_{o \sim \mathcal{A}(D)} \left(\frac{\mathbf{Pr}(\mathcal{A}(D) = o)}{\mathbf{Pr}(\mathcal{A}(D') = o)} \right)^{\alpha} \leq \varepsilon$$

Abadi et al. [1] and Mironov [51] have shown that an (α, ε)-RDP algorithm guarantees $\left(\varepsilon + \frac{\log(1/\delta)}{\alpha - 1}, \delta\right)$-differential privacy. Follow-up works [4, 12] provide tighter conversions. We used the conversion in [12] in our experiments.

To answer a query $f(D)$ with ℓ_2 sensitivity L, i.e., $\max_{\text{neighboring } D, D'} \|f(D) - f(D')\|_2 \leq L$, the Gaussian mechanism [22] returns $f(D) + \mathcal{N}(0, L^2\sigma^2)$, which guarantees $\left(\sqrt{1.25 \log(2/\delta)}/\sigma, \delta\right)$-differential privacy [20, 22] and $(\alpha, \alpha/2\sigma^2)$-RDP [51].

DP-SGD and Privacy Amplification: Differentially-private stochastic gradient descent (DP-SGD) is a common algorithm to solve private optimization problems. The basic idea is to enforce a bounded ℓ_2 norm of individual gradient, and add Gaussian noise to the gradients used in SGD updates. Specifically, consider a dataset $D = \{d_1, \ldots, d_n\}$ and an objective function of the form $\sum_{i=1}^{n} \ell(\theta; d_i)$ for some loss function ℓ. DP-SGD uses an update rule

$$\theta_{t+1} \leftarrow \theta_t - \frac{\eta}{|\mathcal{B}|} \left(\sum_{i \in \mathcal{B}} \mathsf{clip}\left(\nabla_\theta \ell(\theta_t; d_i), L\right) + \mathcal{N}(0, L^2\sigma^2) \right)$$

where $\mathsf{clip}\,(v, L)$ projects v to the ℓ_2-ball of radius L, and $\mathcal{B} \subseteq [n]$ represents a mini-batch of data.

Using the analysis of the Gaussian mechanism, we know that such an update step guarantees $(\alpha, \alpha/2\sigma^2)$-RDP with respect to the mini-batch $\mathcal{B}$. By parallel composition, running one epoch with disjoint mini-batches guarantees $(\alpha, \alpha/2\sigma^2)$-RDP. On the other hand, previous works [1, 6, 69] has shown that if $\mathcal{B}$ is chosen uniformally at random from $[n]$, or if we use poisson sampling to collect a batch of samples $\mathcal{B}$, then one step would guarantee $\left(\alpha, O\left(\alpha/2\sigma^2 \cdot (|\mathcal{B}|/n)^2\right)\right)$-RDP.

Tree-based Aggregation: Consider the problem of privately releasing prefix sum of a data stream, i.e., given a stream $D = (d_1, d_2, \ldots, d_T)$ such that each $d_i \in \mathbb{R}^p$ has ℓ_2 norm bounded by L, we aim to release $s_t = \sum_{i=1}^{t} d_i$ for all $t \in [1, T]$ under differential privacy. Chan et al. [14], Dwork et al. [23] propose a tree-based aggregation algorithm to solve this problem. Consider a complete binary tree $\mathcal{T}$ with leaf nodes as d_1 to d_T, and internal nodes as the sum of all

leaf nodes in its subtree. To release the exact prefix sum s_t, we only need to sum up $O(\log(t))$ nodes. To guarantee differential privacy for releasing the tree $\mathcal{T}$, since any d_i appears in $\log(T)$ nodes in $\mathcal{T}$, using composition, we can add Gaussian noise of standard deviation of the order $L\sqrt{\log(T)\log(1/\delta)}/\varepsilon$ to guarantee (ε, δ)-differential privacy.

Smith and Thakurta [63] used this aggregation algorithm to build a nearly optimal algorithms for private online learning. One important aspect of Smith and Thakurta [63] result is that it showed the privacy guarantee holds even for *adaptively chosen sequences* $\{d_t\}_{t=1}^T$, which is crucial for model training tasks.

3 Private Follow-The-Regularized-Leader

In this section, we provide the formal description of the DP-FTRL algorithm (Algorithm 1) and its privacy analysis. We then show that a variant of differentially private stochastic gradient descent (DP-SGD) [6, 65] can be viewed of as an instantiation of DP-FTRL under appropriate choice of learning rate.

Critically, *our privacy guarantees for DP-FTRL hold when the data D are processed in an arbitrary (even adversarily chosen) order,* and do not depend on the convexity of the loss functions. The utility guarantees, i.e., the regret and the excess risk guarantees require convex losses (i.e., $\ell(\cdot;\cdot)$ is convex in the first parameter). In the presentation below, we assume differentiable losses for brevity. The arguments extend to non-differentiable convex losses via standard use of sub-differentials [32, 62].

3.1 Algorithm Description

The main idea of DP-FTRL is based on three observations: i) For online convex optimization, to bound the regret, for a given loss function $\ell(\theta; d_t)$ (i.e., the loss at time step t), it suffices for the algorithm to operate on a linearization of the loss at θ_t (the model output at time step t): $\widetilde{\ell}(\theta; d_t) = \langle \nabla_\theta \ell(\theta_t; d_t), \theta - \theta_t \rangle$, ii) Under appropriate choice of λ, optimizing for $\theta_{t+1} = \arg\min_{\theta \in \mathcal{C}} \sum_{i=1}^{t} \widetilde{\ell}(\theta; d_t) + \frac{\lambda}{2} \|\theta\|_2^2$ over $\theta \in \mathcal{C}$ gives a good model at step $t+1$, and iii) For all $t \in [n]$, one can privately keep track of $\sum_{i=1}^{t} \widetilde{\ell}(\theta; d_t)$ using the now standard *tree aggregation protocol* [14, 23]. While a variant of this idea was used in [63] under the name of *follow-the-approximate-leader*, one key difference is that they used a quadratic approximation of the regularized loss, i.e., $\ell(\theta; d_t) + \frac{\lambda}{t} \|\theta\|_2^2$. This formulation results in a more complicated algorithm, sub-optimal regret analysis, and failure to maintain structural properties (like sparsity) introduced by composite losses [18, 44, 46, 72].

Algorithm 1 $\mathcal{A}_{\mathsf{FTRL}}$: Differentially Private Follow-The-Regularized-Leader (DP-FTRL)

Require: Data set: $D = [d_1, \cdots, d_n]$ arriving in a stream, in an arbitrary order; constraint set: $\mathcal{C}$, noise scale: σ, regularization parameter: λ, clipping norm: L.

1: $\theta_1 \leftarrow \arg\min_{\theta \in \mathcal{C}} \frac{\lambda}{2} \|\theta\|_2^2$. **Output** θ_1.
2: $\mathcal{T} \leftarrow \texttt{InitializeTree}(n, \sigma^2, L)$.
3: **for** $t \in [n]$ **do**
4: Let $\nabla_t \leftarrow \mathsf{clip}(\nabla_\theta \ell(\theta_t; d_t), L)$, where $\mathsf{clip}(\boldsymbol{v}, L) = \boldsymbol{v} \cdot \min\left\{\frac{L}{\|\boldsymbol{v}\|_2}, 1\right\}$.
5: $\mathcal{T} \leftarrow \texttt{AddToTree}(\mathcal{T}, t, \nabla_t)$.
6: $\boldsymbol{s}_t \leftarrow \texttt{GetSum}(\mathcal{T}, t)$, i.e., estimate $\sum_{i=1}^{t} \nabla_i$ via tree-aggregation protocol.
7: $\theta_{t+1} \leftarrow \arg\min_{\theta \in \mathcal{C}} \langle \boldsymbol{s}_t, \theta \rangle + \frac{\lambda}{2} \|\theta\|_2^2$. **Output** θ_{t+1}.
8: **end for**

Later in the paper, we provide two variants of DP-FTRL (momentum DP-FTRL, and DP-FTRL for least square losses) which will have superior privacy/utility trade-offs for certain problem settings.

DP-FTRL is formally described in Algorithm 1. There are three functions, `InitializeTree`, `AddToTree`, `GetSum`, that correspond to the tree-aggregation algorithm. At a high-level, `InitializeTree` initializes the tree

data structure $\mathcal{T}$, `AddToTree` allows adding a new gradient ∇_t to $\mathcal{T}$, and `GetSum` returns the prefix sum $\sum_{i=1}^{t} \nabla_t$ privately. Please refer to Appendix B.1 for the formal algorithm descriptions.

It can be shown that the error introduced in DP-FTRL due to privacy is dominated by the error in estimating $\sum_{i=1}^{t} \nabla_t$ at each $t \in [n]$. It follows from [63] that for a sequence of (adaptively chosen) vectors $\{\nabla_t\}_{t=1}^{n}$, if we perform $\texttt{AddToTree}\left(\mathcal{T}, t, \nabla_t\right)$ for each $t \in [n]$, then we can write $\texttt{GetSum}\left(\mathcal{T}, t\right) = \sum_{i=1}^{t} \nabla_i + \boldsymbol{b}_t$ where $\boldsymbol{b}_t$ is normally distributed with mean zero, and $\forall t \in [n], \left\|\boldsymbol{b}_t\right\|_2 \leq L\sigma\sqrt{p\lceil\lg(n)\rceil \ln(n/\beta)}$ w.p. at least $1-\beta$.

Momentum Variant: We find that using an additional momentum term $\gamma \in [0, 1]$ with Line 7 in Algorithm 1 replaced by

$$\boldsymbol{v}_t \leftarrow \gamma \cdot \boldsymbol{v}_{t-1} + \boldsymbol{s}_t,\ \theta_{t+1} \leftarrow \arg\min_{\theta \in \mathcal{C}} \langle \boldsymbol{v}_t, \theta \rangle + \frac{\lambda}{2}\|\theta\|_2^2$$

gives superior empirical privacy/utility trade-off compared to the original algorithm when training non-convex models. Throughout the paper, we refer to this variant as momentum DP-FTRL, or DP-FTRLM. Although we do not provide formal regret guarantee for this variant, we conjecture that the superior empirical performance is due to the following reason. The noise added by the tree aggregation algorithm is always bounded by $O(\sqrt{p\ln(1/\delta)} \cdot \ln(n)/\varepsilon)$. However, the noise at time step t and $t+1$ can differ by a factor of $O(\sqrt{\ln n})$. This creates sudden jumps in between the output models comparing to DP-SGD. The momentum can smooth out these jumps.

Privacy analysis: In Theorem 3.1, we provide the privacy guarantee for Algorithm 1 and its momentum variant (with proof in Appendix B.2). In Appendix D, we extend it to multiple passes over the data set D, and batch sizes > 1.

Theorem 3.1 (Privacy guarantee)**.** *If $\left\|\nabla_\theta \ell(\theta; d)\right\|_2 \leq L$ for all $d \in \mathcal{D}$ and $\theta \in \mathcal{C}$, then Algorithm 1 (and its momentum variant) guarantees $\left(\alpha, \frac{\alpha\lceil\lg(n)\rceil}{2\sigma^2}\right)$-Rényi differential privacy, where n is the number of samples in D. Setting $\sigma = \frac{\sqrt{2\lceil\lg(n)\rceil \ln(1/\delta)}}{\varepsilon}$, one can guarantee (ε, δ)-differential privacy, for $\varepsilon \leq 2\ln(1/\delta)$.*

3.2 Comparing Noise in DP-SGD and DP-FTRL

In this section, we use the equivalence of non-private SGD and FTRL [46] to establish equivalence between a variant of noisy-SGD and DP-FTRL, and hence make DP-SGD and DP-FTRl comparable.

Let $D = \{d_1, \ldots, d_n\}$ be the data set of size n. Consider a general noisy-SGD algorithm with update rule $\theta_{t+1} \leftarrow \theta_t - \eta \cdot \left(\nabla_\theta \ell\left(\theta_t; d_t\right) + \boldsymbol{a}_t\right)$, where η is the learning rate and $\boldsymbol{a}_t$ is some random noise. DP-SGD can be viewed as a special case, where d_t is sampled uniformly at random from D and $\boldsymbol{a}_t$ is drawn i.i.d. from $\mathcal{N}\left(0, \widetilde{O}\left(\frac{L^2}{n\varepsilon^2}\right)\right)$. If we expand the recursive relation, we can see that the total amount of noise added to the estimation of θ_{t+1} is $\eta \sum_{i=1}^{t} \boldsymbol{a}_t = \mathcal{N}\left(0, \widetilde{O}\left(\frac{\eta^2 L^2 t}{n\varepsilon^2}\right)\right)$. Let $\boldsymbol{b}_t$ be the noise added by the tree-aggregation algorithm at time step t of Algorithm $\mathcal{A}_{\mathsf{FTRL}}$. We can show that DP-FTRL can be written in the same form as in the above general noisy-SGD formula, where i) the noise $\boldsymbol{a}_t = \boldsymbol{b}_t - \boldsymbol{b}_{t-1}$, ii) the data samples d_t's are drawn in sequence from D, and iii) the learning rate η is set to be $\frac{1}{\lambda}$, where λ is the regularization parameter in Algorithm $\mathcal{A}_{\mathsf{FTRL}}$. In this variant of noisy SGD, the total noise added to the model is $\boldsymbol{b}_t = \mathcal{N}\left(0, \widetilde{O}\left(\frac{\eta^2 \cdot L^2}{\varepsilon^2}\right)\right)$.

Under the same form of the update rule, we can roughly (as the noise is not independent in the DP-FTRL case) compare the two algorithms. When $t = \Omega(n)$, the noise of DP-SGD *with amplification* matches that of DP-FTRL up to factor of $\mathsf{polylog}\,(n)$. As a result, we expect (and as corroborated by the population risk guarantees and experiments) sampled DP-SGD and DP-FTRL to perform similarly. (In Appendix B.3 we provide a formal equivalence.)

4 Regret and Population Risk Guarantees

In this section we consider the setting when loss function ℓ is convex in its first parameter, and provide for DP-FTRL: i) Adversarial regret guarantees for general convex losses, ii) Tighter stochastic regret guarantees for least-squares and

linear losses, and iii) Population risk guarantees via online-to-batch conversion. All our guarantees are high-probability over the randomness of the algorithm, i.e., w.p. at least $1-\beta$, the error only depends on $\mathsf{polylog}(1/\beta)$.

4.1 Adversarial Regret for (Composite) Losses

The theorem here gives a regret guarantee for Algorithm 1 against a *fully adaptive* [62] adversary who chooses the loss function $\ell(\theta; d_t)$ based on $[\theta_1, \ldots, \theta_t]$, but without knowing the internal randomness of the algorithm. See Appendix C.1 for a more general version of Theorem 4.1, and its proof.

Theorem 4.1 (Regret guarantee)**.** *Let θ be any model in $\mathcal{C}$, $[\theta_1, \ldots, \theta_n]$ be the outputs of Algorithm $\mathcal{A}_{\mathsf{FTRL}}$ (Algorithm 1), and let L be a bound on the ℓ_2-Lipschitz constant of the loss functions. Setting λ optimally and plugging in the noise scale σ from Theorem 3.1 to ensure (ε, δ)-differential privacy, we have that for any $\theta^* \in \mathcal{C}$, w.p. at least $1-\beta$ over the randomness of $\mathcal{A}_{\mathsf{FTRL}}$, the regret*

$$R_D(\mathcal{A}_{\mathsf{FTRL}}; \theta^*) = O\left(L \left\|\theta^*\right\|_2 \cdot \left(\frac{1}{\sqrt{n}} + \sqrt{\frac{p^{1/2} \ln^2(1/\delta) \ln(1/\beta)}{\varepsilon n}}\right)\right).$$

Extension to composite losses: Composite losses [18, 44, 46] refer to the setting where in each round, the algorithm is provided with a function $f_t(\theta) = \ell(\theta; d_t) + r_t(\theta)$ with $r_t : \mathcal{C} \to \mathbb{R}^+$ being a convex regularizer that does not depend on the data sample d_t. The ℓ_1-regularizer, $r_t(\theta) = \|\theta\|_1$, is perhaps the most important practical example, playing a critical role in high-dimensional statistics (e.g., in the LASSO method) [10], as well as for applications like click-through-rate (CTR) prediction where very sparse models are needed for efficiency [48]. In order to operate on composite losses, we simply replace Line 7 of Algorithm $\mathcal{A}_{\mathsf{FTRL}}$ with

$$\theta_{t+1} \leftarrow \arg\min_{\theta \in \mathcal{C}} \langle \boldsymbol{s}_t, \theta \rangle + \sum_{i=1}^{t} r_i(\theta) + \frac{\lambda}{2} \|\theta\|_2^2,$$

which can be solved in closed form in many important cases such as ℓ_1 regularization. We obtain Corollary 4.2, analogous to [46, Theorem 1] in the non-private case. We do not require any assumption (e.g., Lipschitzness) on the regularizers beyond convexity since we *only linearize the losses* in Algorithm $\mathcal{A}_{\mathsf{FTRL}}$. It is worth mentioning that [63] is fundamentally incompatible with this type of guarantee.

Corollary 4.2. *Let θ be any model in $\mathcal{C}$, $[\theta_1, \ldots, \theta_n]$ be the outputs of Algorithm $\mathcal{A}_{\mathsf{FTRL}}$ (Algorithm 1), and L be a bound on the ℓ_2-Lipschitz constant of the loss functions. W.p. at least $1-\beta$ over the randomness of the algorithm, for any $\theta^* \in \mathcal{C}$, assuming $\mathbf{0} \in \mathcal{C}$, we have:*

$$R_D(\mathcal{A}_{\mathsf{FTRL}}; \theta^*) \leq \frac{L\sigma\sqrt{p\lceil \lg n \rceil \ln(n/\beta)} + L^2}{\lambda} + \frac{\lambda}{2n} \left\|\theta^*\right\|_2^2 + \frac{1}{n} \sum_{t=1}^{n} r_t(\theta^*).$$

4.2 Stochastic Regret for Least-squared Losses

In this setting, for each data sample $d_i = (\mathbf{x}_i, y_i)$ (with $\mathbf{x}_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$) in the data set $D = \{d_1, \ldots, d_n\}$, the corresponding loss takes the least-squares form[5]: $\ell(\theta; d_i) = (y_i - \langle \mathbf{x}_i, \theta \rangle)^2$. We also assume that each data sample d_i is drawn i.i.d. from some fixed distribution τ.

A straightforward modification of DP-FTRL, $\mathcal{A}_{\mathsf{FTRL\text{-}LS}}$ (given as Algorithm 2 in Appendix C.2), achieves the following guarantee.

[5] A similar argument as in Theorem 4.3 can be used in the setting where the loss functions are linear, $\ell(\theta; d) = \langle \theta, d \rangle$ with $d \in \mathbb{R}^p$ and $\|d\|_2 \leq L$.

Theorem 4.3 (Stochastic regret for least-squared losses)**.** *Let* $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\} \in \mathcal{D}^n$ *be a data set drawn i.i.d. from* τ*, let* $L = \max_{\mathbf{x} \in \mathcal{D}} \|\mathbf{x}\|_2$*, and let* $\max_{y \sim \mathcal{D}} |y| \leq 1$*. Let* $\theta^* \in \mathcal{C}$*,* $\mu = \max_{\theta \in \mathcal{C}} \|\theta\|_2$*, and* $\rho = \max\{\mu, \mu^2\}$*. Then* $\mathcal{A}_{\textsf{FTRL-LS}}$ *provides* (ε, δ)*-differentially privacy while outputting* $[\theta_1, \ldots, \theta_n]$ *s.t. w.p. at least* $1 - \beta$ *for any* $\theta^* \in \mathcal{C}$*,* $\mathbb{E}_D \left[R_D(\mathcal{A}_{\textsf{FTRL-LS}}; \theta^*)\right] = O\left(L^2 \rho^2 \left(\sqrt{\frac{\ln(n)}{n}} + \frac{\sqrt{p \ln^5(n/\beta) \cdot \ln(1/\delta)}}{\varepsilon n}\right)\right)$.

The arguments of [3] can be extended to show a similar regret guarantee *in expectation only*, whereas ours is a high-probability guarantee.

4.3 Excess Risk via Online-to-Batch Conversion

Using the online-to-batch conversion [13, 61], from Theorem 4.1, we can obtain a population risk guarantee $O\left(\left(\sqrt{\frac{\ln(1/\beta)}{n}} + \sqrt{\frac{p^{1/2} \ln^2(1/\delta) \ln(1/\beta)}{\varepsilon n}}\right)\right)$, where β is the failure probability. (See Appendix C.3 for a formal statement.) For least squares and linear losses, using the regret guarantee in Theorem 4.3 and online-to-batch conversion, one can actually achieve the optimal population risk (up to logarithic factors) $O\left(\sqrt{\frac{\ln(n) \ln(1/\beta)}{n}} + \frac{\sqrt{p \ln^5(n/\beta) \cdot \ln(1/\delta)}}{\varepsilon n}\right)$.

5 Empirical Evaluation

We provide an empirical evaluation of DP-FTRL (Algorithm 1) on four benchmark data sets, and compare its performance with the state-of-the-art DP-SGD on three different axes: (1) **Privacy**, measured as an (ε, δ)-DP guarantee on the mechanism, (2) **Utility**, measured as (expected) test set accuracy for the final trained model under the DP guarantee, and (3) **Computation cost**, which we measure in terms of mini-batch size and number of training iterations.

First, we evaluate the privacy/utility trade-offs provided by each technique at fixed computation costs. Second, we evaluate the privacy/computation trade-offs each technique can provide at fixed utility targets. A natural application for this is distributed frameworks such as FL, where the privacy budget and a desired utility threshold can be fixed, and the goal is to satisfy both constraints with the least computation. Computational cost is of critical importance in FL, as it can get challenging to find available clients with increasing mini-batch size and/or number of training rounds.

We show the following results: (1) DP-FTRL provides superior privacy/utility trade-offs than unamplified DP-SGD, (2) For a modest increase in computation cost, DP-FTRL (that does not use any privacy amplification) can match the privacy/utility trade-offs of amplified DP-SGD for all privacy regimes, and further (3) For regimes with large privacy budgets, DP-FTRL achieves higher accuracy than amplified DP-SGD even at the same computation cost, (4) For realistic data set sizes, DP-FTRL can provide superior privacy/computation trade-offs compared to DP-SGD.

5.1 Experimental Setup

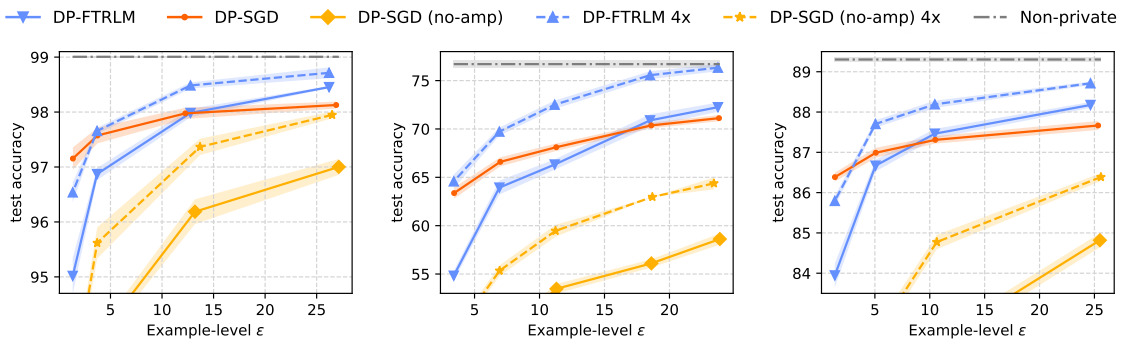
Datasets: We conduct our evaluation on three image classification tasks, MNIST [43], CIFAR-10 [42], EMNIST (ByMerge split) [16]; and a next word prediction task on StackOverflow data set [53]. Since StackOverflow is naturally keyed by users, we assume training in a federated learning setting, i.e., using the Federated Averaging optimizer for training over users in StackOverflow. The privacy guarantee is thus user-level, in contrast to the example-level privacy for the other three datasets (see Definition 1.1).

For all experiments with DP, we set the privacy parameter δ to 10^{-5} on MNIST and CIFAR-10, and 10^{-6} on EMNIST and StackOverflow, s.t. $\delta < n^{-1}$, where n is the number of users in StackOverflow (or the number of examples in the other data sets).

Model Architectures: For all the image classification tasks, we use small convolutional neural networks as in prior work [55]. For StackOverflow, we use the one-layer LSTM network described in [58]. See Appendix E.1 for more details.

Optimizers: We consider DP-FTRL with mini-batch model updates, and multiple epochs. We provide a privacy analysis for both the extensions in Appendix D. We also consider its momentum variant DP-FTRLM. We find that DP-FTRLM with momentum 0.9 always outperforms DP-FTRL. Similarly, for DP-SGD [31], we consider its momentum

variant (DP-SGDM), and report the best-performing variant in each task. See Appendix E.2 for a comparison of the two optimizers for both techniques.



(a) MNIST (b) CIFAR-10 (c) EMNIST (ByMerge)

Figure 1: Privacy/accuracy trade-offs for DP-SGD (private baseline), DP-SGD without amplification (label "DP-SGD (no-amp)"), and DP-FTRLM on MNIST (mini-batch size 250), CIFAR-10 (mini-batch size 500), and EMNIST (mini-batch size 500). "4x" in the label denotes four times computation cost (by increasing batch size four times). Results for "DP-SGD 4x" are deferred to Appendix F.

5.2 Privacy/Utility Trade-offs with Fixed Computation

In Figure 1, we show accuracy / privacy tradeoffs (by varying the noise multiplier) at fixed computation costs. Since both DP-FTRL and DP-SGD require clipping gradients from each sample and adding noise to the aggregated update in each iteration, we consider the number of iterations and the minibatch size as a proxy for computation cost. For each experiment, we run five independent trials, and plot the mean and standard deviation of the final test accuracy at different privacy levels. We provide details of hyperparameter tuning for all the techniques in Appendix F.1.

DP-SGD is the state-of-the-art technique used for private deep learning, and amplification by subsampling (or shuffling) forms a crucial component in its privacy analysis. Thus, we take amplified DP-SGD (or its momentum variant when performance is better) at a fixed computation cost as our baseline (shown as the red lines). We fix the (samples in mini-batch, training iterations) to (250, 4800) for MNIST, (500, 10000) for CIFAR-10, and (500, 69750) for EMNIST. Our goal is to achieve equal or better tradeoffs while processing data in an arbitrary order (that is, without relying on any amplification).

DP-SGD without any privacy amplification (labelled "DP-SGD (no-amp)") cannot achieve this: For all the data sets, we find that the accuracy with DP-SGD (no-amp) at the highest ε in Figure 1 is worse than the accuracy of the DP-SGD baseline even at its lowest ε. Further, if we increase the computation by four times (increasing the mini-batch size by a factor of four), the privacy/utility trade-offs of "DP-SGD (no-amp) 4x" are still substantially worse than the private baseline.[6]

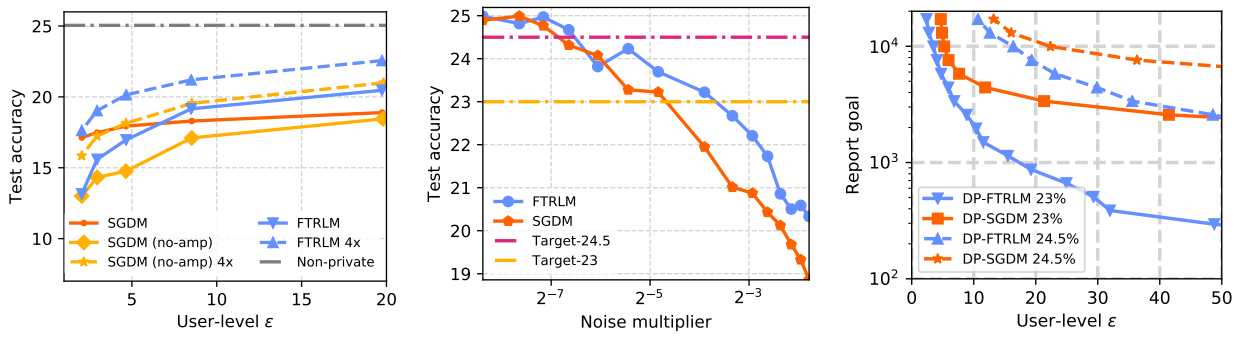
For DP-FTRLM (blue) at the same computation cost as our DP-SGD baseline, as the privacy parameter ε increases, the relative performance of DP-FTRLM improves for each data set, even outperforming the baseline for larger values of ε. Further, if we increase the batch size by four times for DP-FTRLM, its privacy-utility trade-off almost always matches or outperforms the amplified DP-SGD baseline, answering the primary question of this paper in the affirmative. In particular, for CIFAR-10 (Figure 1b), "DP-FTRLM 4x" provides superior performance than the DP-SGD baseline even for the lowest ε.

We observe similar results for StackOverflow with user-level DP in Figure 2a. We fix the computation cost to 100 clients per round (also referred to as the report goal), and 1600 training rounds. DP-SGDM (or more precisely in this case, DP-FedAvg with server momentum) is again our baseline (red). For DP-SGDM without privacy amplification (DP-SGDM no-amp), the privacy/accuracy trade-off never matches that of the DP-SGD baseline, and gets significantly

[6]For completeness, we provide plots with the full performance of DP-SGD (no-amp), DP-SGD (no-amp) 4x, and DP-SGD 4x, in Appendix F.2.

worse for lower ε. With a 4x increase in report goal, DP-SGDM no-amp nearly matches the privacy/utility trade-off of the DP-SGD baseline, outperforming it for larger ε.

For DP-FTRLM, with the same computation cost as the DP-SGD baseline, it outperforms the baseline for the larger ε, whereas for the four-times increased report goal, it provides a strictly better privacy/utility trade-off. We conclude DP-FTRL provides superior privacy/utility trade-offs than unamplified DP-SGD, and for a modest increase in computation cost, it can match the performance of DP-SGD, without the need for privacy amplification.



(a) (b) (c)

Figure 2: (a) Accuracy on StackOverflow under different privacy epsilon by varying noise multiplier and batch sizes. (b) Test accuracy of DP-SGDM and DP-FTRLM with various noise multipliers for StackOverflow. (c) Relationship between user-level privacy ε (when $\delta \approx 1/\text{population}$) and computation cost (report goal) for two fixed accuracy targets (see legend) on the StackOverflow data set.

5.3 Privacy/Computation Trade-offs with Fixed Utility

For a sufficiently large data set / population, better privacy vs. accuracy trade-offs can essentially always be achieved at the cost of increased computation. Thus, in this section we slice the privacy/utility/computation space by fixing utility (accuracy) targets, and evaluating how much computation (report goal) is necessary to achieve different ε for StackOverflow. Our non-private baseline achieves an accuracy of 25.15%, and we fix 24.5% (2.6% relative loss) and 23% (8.6% relative loss) as our accuracy targets. Note that from the accuracy-privacy trade-offs presented in Figure 2a, achieving even 23% for either DP-SGD or DP-FTRL will result in a very large privacy ε for the report goals considered there.

For each target, we tune hyperparameters (see Appendix G.2 for details) for both DP-SGDM and DP-FTRLM at a fixed computation cost to obtain the maximum noise scale for each technique while ensuring the trained models meet the accuracy target. Specifically, we fix a report goal of 100 clients per round for 1600 training rounds, and tune DP-SGD and DP-FTRL for 15 noise multipliers, ranging from $(0, 0.3)$ for DP-SGD, and $(0, 1.13)$ for DP-FTRL. At this report goal, for noise multiplier 0.3, DP-SGD provides 18.89% accuracy at $\varepsilon \sim 19$, whereas for noise multiplier 1.13 DP-FTRL provides 19.74% accuracy at $\varepsilon \sim 19$. We provide the results in Figure 2b.

Now, for target accuracy 23% and 24.5%, we choose the largest noise multiplier for each technique that results in the trained model achieving the accuracy target. For accuracies (23%, 24.5%), we select noise multipliers (0.015, 0.007) for DP-SGDM, and (0.268, 0.067) for DP-FTRLM, respectively. This data allows us to evaluate the privacy/computation trade-offs for both the techniques, assuming the accuracy stays constant as we scale up the noise and report goal together (maintaining a constant signal-to-noise ratio while improving ε). This assumption was introduced and validated by [49], which showed that keeping the clipping norm bound, training rounds, and the scale of the noise added to the model update constant, increasing the report goal does not change the final model accuracy. In Appendix G.1, we independently corroborate this effect for both DP-SGD and DP-FTRL on StackOverflow.

We plot the results in Figure 2c. For both the accuracy targets, DP-FTRLM achieves any privacy $\varepsilon \in (0, 50)$ at a lower computational cost than DP-SGDM. In Appendix G.3, we provide a similar plot for a hypothetically larger

population, where we see that DP-FTRLM provides superior performance than DP-SGDM for most of the considered privacy regimes.

6 Conclusion

In this paper we introduce the DP-FTRL algorithm, which we show to have the tightest known regret guarantees under DP, and have the best known excess population risk guarantees for a single pass algorithm on non-smooth convex losses. For linear and least-squared losses, we show DP-FTRL actually achieves the optimal population risk. Furthermore, we show on benchmark data sets that DP-FTRL, which does not rely on any privacy amplification, can outperform amplified DP-SGD at large values of ε, and be competitive to it for all ranges of ε for a modest increase in computation cost (batch size). This work leaves two main open questions: i) Can DP-FTRL achieve the optimal excess population risk for all convex losses in a single pass?, and ii) Can one tighten the empirical gap between DP-SGD and DP-FTRL at smaller values of ε, possibly via a better estimator of the gradient sums from the tree data structure?

Acknowledgements

We would like to thank Borja Balle and Satyen Kale for the helpful discussions through the course of this project.

References

[1] M. Abadi, A. Chu, I. J. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *Proc. of the 2016 ACM SIGSAC Conf. on Computer and Communications Security (CCS'16)*, pages 308–318, 2016.

[2] J. Abernethy, Y. H. Jung, C. Lee, A. McMillan, and A. Tewari. Online learning via the differential privacy lens. In *NeurIPS*, 2019.

[3] N. Agarwal and K. Singh. The price of differential privacy for online learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 32–40, 2017.

[4] S. Asoodeh, J. Liao, F. P. Calmon, O. Kosut, and L. Sankar. A better bound gives a hundred rounds: Enhanced privacy guarantees via f-divergences. In *2020 IEEE International Symposium on Information Theory (ISIT)*, pages 920–925. IEEE, 2020.

[5] B. Balle, P. Kairouz, B. McMahan, O. D. Thakkar, and A. Thakurta. Privacy amplification via random check-ins. *Advances in Neural Information Processing Systems*, 33, 2020.

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

[7] R. Bassily, V. Feldman, K. Talwar, and A. Thakurta. Private stochastic convex optimization with optimal rates. In *Advances in Neural Information Processing Systems*, pages 11279–11288, 2019.

[8] R. Bassily, V. Feldman, K. Talwar, and A. G. Thakurta. Private stochastic convex optimization with optimal rates. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 11279–11288, 2019.

[9] R. Bassily, V. Feldman, C. Guzmán, and K. Talwar. Stability of stochastic gradient descent on nonsmooth convex losses. *arXiv preprint arXiv:2006.06914*, 2020.

[10] P. Bhlmann and S. van de Geer. *Statistics for High-Dimensional Data: Methods, Theory and Applications*. Springer Publishing Company, Incorporated, 2011. ISBN 3642201911.

[11] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecny, S. Mazzocchi, H. B. McMahan, et al. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*, 2019.

[12] C. Canonne, G. Kamath, and T. Steinke. The discrete gaussian for differential privacy. *arXiv preprint arXiv:2004.00010*, 2020.

[13] N. Cesa-Bianchi, A. Conconi, and C. Gentile. On the genomeralization ability of on-line learning algorithms. In *Advances in neural information processing systems*, pages 359–366, 2002.

[14] T.-H. H. Chan, E. Shi, and D. Song. Private and continual release of statistics. *ACM Trans. on Information Systems Security*, 14(3):26:1–26:24, Nov. 2011.

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

[16] G. Cohen, S. Afshar, J. Tapson, and A. V. Schaik. Emnist: Extending mnist to handwritten letters. *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017. doi: 10.1109/ijcnn.2017.7966217.

[17] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[18] J. C. Duchi, S. Shalev-Shwartz, Y. Singer, and A. Tewari. Composite objective mirror descent. In *COLT*, pages 14–26. Citeseer, 2010.

[19] J. C. Duchi, M. I. Jordan, and M. J. Wainwright. Local privacy and statistical minimax rates. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 429–438. IEEE Computer Society, 2013. doi: 10.1109/FOCS.2013.53. URL `https://doi.org/10.1109/FOCS.2013.53`.

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

[21] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our data, ourselves: Privacy via distributed noise generation. In *Advances in Cryptology—EUROCRYPT*, pages 486–503, 2006.

[22] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Proc. of the Third Conf. on Theory of Cryptography (TCC)*, pages 265–284, 2006. URL `http://dx.doi.org/10.1007/11681878_14`.

[23] C. Dwork, M. Naor, T. Pitassi, and G. N. Rothblum. Differential privacy under continual observation. In *Proc. of the Forty-Second ACM Symp. on Theory of Computing (STOC'10)*, pages 715–724, 2010.

[24] 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 *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2468–2479. SIAM, 2019.

[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. *CoRR*, abs/2001.03618, 2020.

[26] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 211–222, 2003.

[27] Facebook. Introducing opacus: A high-speed library for training pytorch models with differential privacy, 2020.

[28] V. Feldman, I. Mironov, K. Talwar, and A. Thakurta. Privacy amplification by iteration. In *59th Annual IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 521–532, 2018.

[29] V. Feldman, T. Koren, and K. Talwar. Private stochastic convex optimization: Optimal rates in linear time. In *Proc. of the Fifty-Second ACM Symp. on Theory of Computing (STOC'20)*, 2020.

[30] V. Feldman, A. McMillan, and K. Talwar. Hiding among the clones: A simple and nearly optimal analysis of privacy amplification by shuffling. *arXiv preprint arXiv:2012.12803*, 2020.

[31] Google. Tensorflow-privacy. `https://github.com/tensorflow/privacy`, 2019.

[32] E. Hazan. Introduction to online convex optimization. *arXiv preprint arXiv:1909.05207*, 2019.

[33] E. Hazan and S. Kale. Beyond the regret minimization barrier: optimal algorithms for stochastic strongly-convex optimization. *The Journal of Machine Learning Research*, 15(1):2489–2512, 2014.

[34] R. Iyengar, J. P. Near, D. Song, O. Thakkar, A. Thakurta, and L. Wang. Towards practical differentially private convex optimization. In *2019 IEEE Symposium on Security and Privacy (SP)*, 2019.

[35] M. Jagielski, J. Ullman, and A. Oprea. Auditing differentially private machine learning: How private is private sgd? *arXiv preprint arXiv:2006.07709*, 2020.

[36] P. Jain and A. G. Thakurta. (near) dimension independent risk bounds for differentially private learning. In *International Conference on Machine Learning*, pages 476–484, 2014.

[37] P. Jain, P. Kothari, and A. Thakurta. Differentially private online learning. In *Proc. of the 25th Annual Conf. on Learning Theory (COLT)*, volume 23, pages 24.1–24.34, June 2012.

[38] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

[39] A. Kalai and S. Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307, 2005.

[40] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. D. Smith. What can we learn privately? In *49th Annual IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 531–540, 2008.

[41] D. Kifer, A. Smith, and A. Thakurta. Private convex empirical risk minimization and high-dimensional regression. In *Conference on Learning Theory*, pages 25–1, 2012.

[42] A. Krizhevsky. Learning multiple layers of features from tiny images, 2009.

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

[44] B. McMahan. Follow-the-regularized-leader and mirror descent: Equivalence theorems and l1 regularization. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 525–533, 2011.

[45] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, pages 1273–1282, 2017. URL `http://proceedings.mlr.press/v54/mcmahan17a.html`.

[46] H. B. McMahan. A survey of algorithms and analysis for adaptive online learning. *Journal of Machine Learning Research*, 18(90):1–50, 2017. URL `http://jmlr.org/papers/v18/14-428.html`.

[47] H. B. McMahan and M. Streeter. Adaptive bound optimization for online convex optimization. In *Proceedings of the 23rd Annual Conference on Learning Theory (COLT)*, 2010.

[48] H. B. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin, et al. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1222–1230, 2013.

[49] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang. Learning differentially private recurrent language models. *arXiv preprint arXiv:1710.06963*, 2017.

[50] H. B. McMahan, G. Andrew, U. Erlingsson, S. Chien, I. Mironov, N. Papernot, and P. Kairouz. A general approach to adding differential privacy to iterative training procedures. *arXiv preprint arXiv:1812.06210*, 2018.

[51] I. Mironov. Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 263–275. IEEE, 2017.

[52] M. Nasr, S. Song, A. Thakurta, N. Papernot, and N. Carlini. Adversary instantiation: Lower bounds for differentially private machine learning. In *IEEE S and P (Oakland)*, 2021.

[53] S. Overflow. The Stack Overflow Data, 2018. `https://www.kaggle.com/stackoverflow/stackoverflow`.

[54] N. Papernot, S. Chien, S. Song, A. Thakurta, and U. Erlingsson. Making the shoe fit: Architectures, initializations, and tuning for learning with privacy, 2020. URL `https://openreview.net/forum?id=rJg851rYwH`.

[55] N. Papernot, A. Thakurta, S. Song, S. Chien, and Ú. Erlingsson. Tempered sigmoid activations for deep learning with differential privacy. *arXiv preprint arXiv:2007.14191*, 2020.

[56] V. Pichapati, A. T. Suresh, F. X. Yu, S. J. Reddi, and S. Kumar. Adaclip: Adaptive clipping for private sgd. *arXiv preprint arXiv:1908.07643*, 2019.

[57] S. Ramaswamy, O. Thakkar, R. Mathews, G. Andrew, H. B. McMahan, and F. Beaufays. Training production language models without memorizing user data, 2020.

[58] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečnỳ, S. Kumar, and H. B. McMahan. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*, 2020.

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

[60] S. Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2):107–194, 2012.

[61] S. Shalev-Shwartz, O. Shamir, N. Srebro, and K. Sridharan. Stochastic convex optimization. In *COLT 2009 - The 22nd Conference on Learning Theory, Montreal, Quebec, Canada, June 18-21, 2009*, 2009. URL `http://www.cs.mcgill.ca/%7Ecolt2009/papers/018.pdf#page=1`.

[62] S. Shalev-Shwartz et al. Online learning and online convex optimization. *Foundations and trends in Machine Learning*, 4(2):107–194, 2011.

[63] A. Smith and A. Thakurta. (nearly) optimal algorithms for private online learning in full-information and bandit settings. In *Advances in Neural Information Processing Systems*, pages 2733–2741, 2013.

[64] C. Song and V. Shmatikov. Auditing data provenance in text-generation models. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 196–206, 2019.

[65] S. Song, K. Chaudhuri, and A. D. Sarwate. Stochastic gradient descent with differentially private updates. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 245–248. IEEE, 2013.

[66] O. Thakkar, G. Andrew, and H. B. McMahan. Differentially private learning with adaptive clipping. *CoRR*, abs/1905.03871, 2019. URL `http://arxiv.org/abs/1905.03871`.

[67] O. Thakkar, S. Ramaswamy, R. Mathews, and F. Beaufays. Understanding unintended memorization in federated learning. *arXiv preprint arXiv:2006.07490*, 2020.

[68] F. Tramèr and D. Boneh. Differentially private learning needs better features (or much more data). In *International Conference on Learning Representations (ICLR)*, 2021.

[69] Y.-X. Wang, B. Balle, and S. P. Kasiviswanathan. Subsampled rényi differential privacy and analytical moments accountant. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1226–1235. PMLR, 2019.

[70] S. L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *J. of the American Statistical Association*, 60(309):63–69, 1965.

[71] X. Wu, F. Li, A. Kumar, K. Chaudhuri, S. Jha, and J. F. Naughton. Bolt-on differential privacy for scalable stochastic gradient descent-based analytics. In S. Salihoglu, W. Zhou, R. Chirkova, J. Yang, and D. Suciu, editors, *Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD*, 2017.

[72] L. Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *The Journal of Machine Learning Research*, 11:2543–2596, 2010.

[73] Y. Zhu and Y.-X. Wang. Poission subsampled rényi differential privacy. In *International Conference on Machine Learning*, pages 7634–7642. PMLR, 2019.

A Other Related Work

Differentially private empirical risk minimization (ERM) and private online learning are well-studied areas in the privacy literature [1, 2, 3, 6, 8, 15, 29, 34, 36, 37, 41, 49, 55, 56, 63, 65, 66, 71][7]. The connection between private ERM and private online learning was first explored in [37], and the idea of using stability induced by differential privacy for designing low-regret algorithms was explored in [2, 3, 39]. To the best of our knowledge, this paper for the first time explores the idea using a purely online learning algorithm for training deep learning models, without relying on any stochasticity in the data for privacy.

B Missing Details from Section 3

B.1 Details of the Tree Aggregation Scheme

In this section we provide the formal details of the tree aggregation scheme used in Algorithm 1 (Algorithm $\mathcal{A}_{\mathsf{FTRL}}$).

1. $\texttt{InitializeTree}\,(n, \sigma^2, L)$: Initialize a complete binary tree $\mathcal{T}$ with $2^{\lceil \lg(n) \rceil}$ leaf nodes, with each node being sampled i.i.d. from $\mathcal{N}(0, L^2\sigma^2 \cdot \mathbb{I}_{p\times p})$.
2. $\texttt{AddToTree}\,(\mathcal{T}, t, \boldsymbol{v})$: Add $\boldsymbol{v}$ to all the nodes along the path to the root of $\mathcal{T}$, starting from t-th leaf node.
3. $\texttt{GetSum}\,(\mathcal{T}, t)$: Let $[\texttt{node}_1, \ldots, \texttt{node}_h]$ be the list of nodes from the t-th leaf node to the root of $\mathcal{T}$, with $\texttt{node}_1$ being the root node and $\texttt{node}_h$ being the leaf node.

[7] This is only a small representative subset of the literature.

(a) Initialize $\boldsymbol{s} \leftarrow \mathbf{0}^p$ and convert t to binary in h bit representation $[b_1, \ldots, b_h]$, with b_1 being the most significant bit.

(b) For each $j \in [h]$, if $b_j = 1$, then add the value in left sibling of $\texttt{node}_j$ to $\boldsymbol{s}$. Here if $\texttt{node}_j$ is the left child, then it is treated as its own left sibling.

(c) Return $\boldsymbol{s}$.

B.2 Proof of Theorem 3.1

Proof. Notice that in Algorithm 1, all accesses to private information is only through the tree data structure $\mathcal{T}$. Hence, to prove the privacy guarantee, it is sufficient to show that for any data set $V = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n\}$ (with each $\|\boldsymbol{v}_i\|_2 \leq L$), the operations on the tree data structure (i.e., the `InitializeTree`, `AddToTree`, `GetSum`) provide the privacy guarantees in the Theorem statement. First, notice that each $\boldsymbol{v}_i$ affects at most $\lceil \lg(n) \rceil$ nodes in the tree $\mathcal{T}$. Additionally, notice that the computation in each node of the tree $\mathcal{T}$ is essentially a summation query. With these two observations, one can use standard properties of Gaussian mechanism [21],[51, Corollary 3], and adaptive RDP composition [51, Proposition1] to complete the proof.

While the original work on tree aggregation [14, 23] did not use either Gaussian mechanism or RDP composition, it is not hard to observe that the translation to the current setting is immediate. □

B.3 Missing details from Section 3.2 (Comparing Noise in DP-SGD and DP-FTRL)

Theorem B.1. *Consider data set $D = \{d_1, \ldots, d_n\}$, model space $\mathcal{C} = \mathbb{R}^p$ and initial model $\theta_0 = \mathbf{0}^p$. For $t \in [n]$, let the update of Noisy-SGD be $\theta_{t+1}^{\textsf{Noisy-SGD}} \leftarrow \theta_t - \eta \cdot \left(\nabla_\theta \ell \left(\theta_t^{\textsf{Noisy-SGD}}; d_t \right) + \boldsymbol{a}_t \right)$, where $\boldsymbol{a}_t$'s are noise random variables. Let the DP-FTRL (Algorithm 1) updates be $\theta_{t+1}^{\textsf{DP-FTRL}} \leftarrow \arg\min_{\theta \in \mathbb{R}^p} \sum_{i=1}^{t} \nabla_\theta \ell \left(\theta_i^{\textsf{DP-FTRL}}; d_i \right) + \langle \boldsymbol{b}_t, \theta \rangle + \frac{1}{2\eta} \|\theta\|_2^2$, where $\boldsymbol{b}_t$'s are the noises added by the tree-aggregation mechanism.*

If we instantiate $\boldsymbol{a}_t = \boldsymbol{b}_t - \boldsymbol{b}_{t-1}$, and $\eta = \frac{1}{\lambda}$, then for all $t \in [n]$, $\theta_t^{\textsf{Noisy-SGD}} = \theta_t^{\textsf{DP-FTRL}}$.

Proof. Consider the non-private SGD and FTRL. Recall that the SGD update is $\theta_{t+1}^{\textsf{SGD}} \leftarrow \theta_t^{\textsf{SGD}} - \eta \nabla_\theta \ell(\theta_t^{\textsf{SGD}}; d_t)$, where η is the learning rate. Opening up the recurrence, we have $\theta_{t+1}^{\textsf{SGD}} \leftarrow \theta_0 - \eta \sum_{i=1}^{t} \nabla_\theta \ell(\theta_i^{\textsf{SGD}}; d_i)$. If $\theta_0^{\textsf{SGD}} = \mathbf{0}^p$, then equivalently $\theta_{t+1}^{\textsf{SGD}} \leftarrow \arg\min_{\theta \in \mathbb{R}^p} \langle \sum_{i=1}^{t} \nabla_\theta \ell(\theta_i^{\textsf{SGD}}; d_i), \theta \rangle + \frac{1}{2\eta} \|\theta\|_2^2$. This is identical to the update rule of the non-private FTRL (i.e., with σ set to 0 in DP-FTRL) with regularization parameter λ set to $\frac{1}{\eta}$.

Now we consider the Noisy-SGD and DP-FTRL. Recall that Noisy-SGD has update rule $\theta_{t+1}^{\textsf{Noisy-SGD}} \leftarrow \theta_t^{\textsf{Noisy-SGD}} - \eta \left(\nabla_\theta \ell(\theta_t^{\textsf{Noisy-SGD}}; d_t) + \boldsymbol{a}_t \right)$, where $\boldsymbol{a}_t$ is the Gaussian noise added at time step t. Similar as before, this rule can be written as

$$\theta_{t+1}^{\textsf{Noisy-SGD}} \leftarrow \arg\min_{\theta \in \mathbb{R}^p} \langle \sum_{i=1}^{t} \nabla_\theta \ell(\theta_i^{\textsf{Noisy-SGD}}; d_i), \theta \rangle + \langle \sum_{i=1}^{t} \boldsymbol{a}_i, \theta \rangle + \frac{1}{2\eta} \|\theta\|_2^2 . \tag{3}$$

The update rule of DP-FTRL can be written as

$$\theta_{t+1}^{\textsf{DP-FTRL}} \leftarrow \arg\min_{\theta \in \mathbb{R}^p} \langle \sum_{i=1}^{t} \nabla_\theta \ell \left(\theta_i^{\textsf{DP-FTRL}}; d_i \right), \theta \rangle + \langle \boldsymbol{b}_t, \theta \rangle + \frac{\lambda}{2} \|\theta\|_2^2 , \tag{4}$$

where $\boldsymbol{b}_t$ is the noise that gets added by the tree-aggregation mechanism at time step $t+1$. If we 1) set $\lambda = \frac{1}{\eta}$, 2) draw data samples sequentially from D in Noisy-SGD, and 3) set $\boldsymbol{a}_t = \boldsymbol{b}_t - \boldsymbol{b}_{t-1}$ so that $\sum_{i=1}^{t} \boldsymbol{a}_t = \boldsymbol{b}_t$, we can establish the equivalence between (3) and (4). This completes the proof. □

C Missing Details from Section 4

C.1 Proof of Theorem 4.1

We first present a more detailed version of Theorem 4.1 and then present its proof.

Theorem C.1 (Regret guarantee (Theorem 4.1 in detail)). *Let $[\theta_1, \ldots, \theta_n]$ be the outputs of Algorithm $\mathcal{A}_{\mathsf{FTRL}}$ (Algorithm 1), and L be a bound on the ℓ_2-Lipschitz constant of the loss functions. W.p. at least $1-\beta$ over the randomness of $\mathcal{A}_{\mathsf{FTRL}}$, the following is true for any $\theta^* \in \mathcal{C}$.*

$$\frac{1}{n}\sum_{t=1}^{n}\ell(\theta_t; d_t) - \frac{1}{n}\sum_{t=1}^{n}\ell(\theta^*; d_t) \leq \frac{L\sigma\sqrt{p\lceil \lg n\rceil \ln(n/\beta)} + L^2}{\lambda} + \frac{\lambda}{2n}\left(\|\theta^*\|_2^2 - \|\theta_1\|_2^2\right)$$

Setting λ optimally and plugging in the noise scale σ from Theorem 3.1 to ensure (ε, δ)-differential privacy, we have

$$R_D(\mathcal{A}_{\mathsf{FTRL}}; \theta^*) = O\left(L\|\theta^*\|_2 \cdot \left(\frac{1}{\sqrt{n}} + \sqrt{\frac{p^{1/2}\ln^2(1/\delta)\ln(1/\beta)}{\varepsilon n}}\right)\right).$$

Proof. Recall that by Algorithm $\mathcal{A}_{\mathsf{FTRL}}$, $\theta_{t+1} \leftarrow \arg\min_{\theta\in\mathcal{C}} \underbrace{\sum_{i=1}^{t}\langle\nabla_i, \theta\rangle + \frac{\lambda}{2}\|\theta\|_2^2 + \langle \boldsymbol{b}_t, \theta\rangle}_{J_t^{\mathsf{priv}}(\theta)}$, where the Gaussian noise $\boldsymbol{b}_t = \boldsymbol{s}_t - \sum_{i=1}^{t}\nabla_i$ for $\boldsymbol{s}_t$ being the output of $\texttt{GetSum}(\mathcal{T}, t)$. By standard concentration of spherical Gaussians, w.p. at least $1-\beta$, $\forall t \in [n]$, $\|\boldsymbol{b}_t\|_2 \leq L\sigma\sqrt{p\lceil\lg(n)\rceil\ln(n/\beta)}$. We will use this bound to control the error introduced due to privacy. Now, consider the optimizer of the non-private objective:

$$\widetilde{\theta}_{t+1} \leftarrow \arg\min_{\theta\in\mathcal{C}} \underbrace{\sum_{i=1}^{t}\langle\nabla_i, \theta\rangle + \frac{\lambda}{2}\|\theta\|_2^2}_{J_t^{\mathsf{np}}(\theta)}, \qquad \text{where } \nabla_t = \nabla\ell(\theta_t; d_t).$$

That is, post-hoc we consider the hypothetical application of non-private FTRL to the same sequence of *linearized* loss functions $f_t(\widetilde{\theta}) = \langle\nabla_t, \widetilde{\theta}\rangle = \langle\nabla\ell(\theta_t; d_t), \widetilde{\theta}\rangle$ seen in the private training run. In the following, we will first bound how much the models output by $\mathcal{A}_{\mathsf{FTRL}}$ deviate from models output by the hypothetical non-private FTRL discussed above. Then, we invoke standard regret bound for FTRL, while accounting for the deviation of the models output by $\mathcal{A}_{\mathsf{FTRL}}$.

To bound $\left\|\widetilde{\theta}_{t+1} - \theta_{t+1}\right\|_2$, we apply Lemma C.2. We set $\phi_1(\theta) = J_t^{\mathsf{np}}(\theta)/\lambda$, $\phi_2(\theta) = J_t^{\mathsf{priv}}(\theta)/\lambda$, and both $\|\cdot\|$ and its dual as the ℓ_2 norm. We thus have $\Psi(\theta) = \langle\boldsymbol{b}_t, \theta\rangle/\lambda$, with $\boldsymbol{b}_t/\lambda$ being its subgradient. Therefore,

$$\left\|\widetilde{\theta}_{t+1} - \theta_{t+1}\right\|_2 \leq \frac{\|\boldsymbol{b}_t\|_2}{\lambda}. \tag{5}$$

Lemma C.2 (Lemma 7 from [46] restated). *Let $\phi_1 : \mathcal{C} \to \mathbb{R}$ be a convex function (defined over $\mathcal{C} \subseteq \mathbb{R}^p$) s.t. $\theta_1 \in \arg\min_{\theta\in\mathcal{C}}\phi_1(\theta)$ exists. Let $\Psi(\theta)$ be a convex function s.t. $\phi_2(\theta) = \phi_1(\theta) + \Psi(\theta)$ is 1-strongly convex w.r.t. $\|\cdot\|$-norm. Let $\theta_2 \in \arg\min_{\theta\in\mathcal{C}}\phi_2(\theta)$. Then for any $\boldsymbol{b}$ in the subgradient of Ψ at θ_1, the following is true: $\|\theta_1 - \theta_2\|_* \leq \|\boldsymbol{b}\|_*$. Here $\|\cdot\|_*$ is the dual-norm of $\|\cdot\|$.*

We can now easily bound the regret. By standard linear approximation "trick" from the online learning literature [32, 60], we have the following. For $\nabla_t = \nabla_\theta \ell(\theta_t; d_t)$,

$$\begin{aligned}\frac{1}{n}\sum_{t=1}^{n}\ell(\theta_t;d_t) - \frac{1}{n}\sum_{t=1}^{n}\ell(\theta^*;d_t) &\le \frac{1}{n}\sum_{t=1}^{n}\langle\nabla_t, \theta_t - \theta^*\rangle \\ &= \frac{1}{n}\sum_{t=1}^{n}\langle\nabla_t, \theta_t - \widetilde{\theta}_t + \widetilde{\theta}_t - \theta^*\rangle \\ &= \underbrace{\frac{1}{n}\sum_{t=1}^{n}\langle\nabla_t, \widetilde{\theta}_t - \theta^*\rangle}_{A} + \underbrace{\frac{1}{n}\sum_{t=1}^{n}\langle\nabla_t, \theta_t - \widetilde{\theta}_t\rangle}_{B}. \end{aligned} \tag{6}$$

One can bound the term A in (6) by [32, Theorem 5.2] and get $A \le \left(\frac{L^2}{\lambda} + \frac{\lambda}{2n}\left(\|\theta^*\|_2^2 - \|\theta_1\|_2^2\right)\right)$. As for term B, using (5) and the concentration on $\boldsymbol{b}_t$ mentioned earlier, we have, w.p. at least $1-\beta$,

$$B \le \frac{1}{n}\sum_{t=1}^{n}\|\nabla_t\|_2 \cdot \left\|\widetilde{\theta}_t - \theta_t\right\|_2 \le \frac{1}{n}\sum_{t=1}^{n} L \cdot \left\|\widetilde{\theta}_t - \theta_t\right\|_2 \le \frac{L\sigma\sqrt{p\lceil\lg n\rceil\ln(n/\beta)}}{\lambda}. \tag{7}$$

Combining (6) and (7), we immediately have the first part of of Theorem 4.1. To prove the second part of the theorem, we just optimize for the regularization parameter λ and plug in the noise scale σ from Theorem 3.1. □

C.2 Additional Details for Section 4.2

In Algorithm 2, we present a version of DP-FTRL for least square loss. In this modified algorithm, the functions `InitializeTreeBias`, `AddToTreeBias`, and `GetSumBias` are identical to `InitializeTree`, `AddToTree`, and `GetSum` respectively in Algorithm 1. The functions `AddToTreeCov`, `AddToTreeCov`, and `GetSumCov` are similar to `InitializeTree`, `AddToTree`, and `GetSum`, except that the p-dimensional vector versions are replaced by $p \times p$-dimensional matrix version, and the noise in `InitializeTreeCov` is initialized by symmetric $p \times p$ Gaussian matrices with each entry drawn i.i.d. from $\mathcal{N}\left(0, L^4\sigma^2\right)$.

Algorithm 2 $\mathcal{A}_{\mathsf{FTRL\text{-}LS}}$: Differentially Private Follow-The-Regularized-Leader (DP-FTRL) for least-squared losses

Require: Data set: $D = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)\}$ arriving in a stream, constraint set: $\mathcal{C}$, noise scale: σ, regularization parameter: λ, upper bound on $\{\|\mathbf{x}_t\|_2\}_{t=1}^n : L$.

$\theta_1 \leftarrow \arg\min_{\theta\in\mathcal{C}} \frac{\lambda}{2}\|\theta\|_2^2$. **Output** θ_1.

$\mathcal{T}_{\mathsf{bias}} \leftarrow$ `InitializeTreeBias` (n, σ^2, L), $\mathcal{T}_{\mathsf{cov}} \leftarrow$ `InitializeTreeCov` (n, σ^2, L^2).

for $t \in [n]$ **do**

  Let $\boldsymbol{v}_t \leftarrow y_t \cdot \mathbf{x}_t$, and $M_t \leftarrow \mathbf{x}_t\mathbf{x}_t^\top$.

  $\mathcal{T}_{\mathsf{bias}} \leftarrow$ `AddToTreeBias` $(\mathcal{T}_{\mathsf{bias}}, t, \boldsymbol{v}_t)$ and $\mathcal{T}_{\mathsf{cov}} \leftarrow$ `AddToTreeCov` $(\mathcal{T}_{\mathsf{cov}}, t, M_t)$.

  $\boldsymbol{s}_t \leftarrow$ `GetSumBias` $(\mathcal{T}_{\mathsf{bias}}, t)$, and $W_t \leftarrow$ `GetSumCov` $(\mathcal{T}_{\mathsf{cov}}, t)$.

  $\theta_{t+1} \leftarrow \arg\min_{\theta\in\mathcal{C}} \left(\theta^\top \cdot W_t \cdot \theta - 2\langle\boldsymbol{s}_t, \theta\rangle\right) + \frac{\lambda}{2}\|\theta\|_2^2$. **Output** θ_{t+1}.

end for

We first present the privacy guarantee of Algorithm 2 in Theorem C.3. Its proof is almost identical to that of Theorem 3.1, except that we need to measure the sensitivity of the covariance matrix in the Frobenius norm.

Theorem C.3 (Privacy guarantee)**.** *If $\|\mathbf{x}\|_2 \le L$ and $|y| \le 1$ for all $(\mathbf{x}, y) \in \mathcal{D}$ and $\theta \in \mathcal{C}$, then Algorithm 1 (Algorithm $\mathcal{A}_{\mathsf{FTRL}}$) satisfies $\left(\alpha, \frac{\alpha\lceil\lg(n)\rceil}{\sigma^2}\right)$-RDP. Correspondingly, by setting $\sigma = \frac{2\sqrt{\lceil\lg(n)\rceil\ln(1/\delta)}}{\varepsilon}$ one can satisfy (ε, δ)-differential privacy guarantee, as long as $\varepsilon \le 2\ln(1/\delta)$.*

In Theorem C.4, we present the regret guarantee for Algorithm 2.

Theorem C.4 (Stochastic regret for least-squared losses)**.** *Let $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\} \in \mathcal{D}^n$ be a data set drawn i.i.d. from τ, with $L = \max_{\mathbf{x}\in\mathcal{D}} \|\mathbf{x}\|_2$ and $\max_{y\sim\mathcal{D}} |y| \le 1$. Let $\mathcal{C}$ be the model space and $\mu = \max_{\theta\in\mathcal{C}} \|\theta\|_2$. Let θ^* be any model in $\mathcal{C}$, and $[\theta_1, \ldots, \theta_n]$ be the outputs of Algorithm $\mathcal{A}_{\textsf{FTRL-LS}}$ (Algorithm 2). Then w.p. at least $1-\beta$ (over the randomness of the algorithm), we have*

$$\mathbb{E}_D\left[R_D(\mathcal{A}_{\textsf{FTRL-LS}}; \theta^*)\right] = \mathbb{E}_D\left[\frac{1}{n}\sum_{t=1}^{n}\left(y_t - \langle \mathbf{x}_t, \theta_t\rangle\right)^2 - \left(y_t - \langle \mathbf{x}_t, \theta^*\rangle\right)^2\right]$$
$$= O\left(\frac{p\ln^2(n)\ln(n/\beta)\sigma^2\cdot\left(L^2 + L^4\mu^2 + L^3\mu\right)}{\lambda n} + \frac{L^4\mu^2}{\lambda} + \frac{\lambda\ln(n)}{n}\cdot\|\theta^*\|_2^2\right).$$

Setting λ optimally and plugging in the noise scale σ from Theorem C.3 to ensure (ε, δ)-differential privacy, we have,

$$\mathbb{E}_D\left[R_D(\mathcal{A}_{\textsf{FTRL-LS}}; \theta^*)\right] = L^2\cdot\|\theta^*\|_2 \cdot O\left(\left(\sqrt{\frac{\mu^2\ln(n)}{n}} + \frac{\sqrt{p\ln^5(n/\beta)\cdot\ln(1/\delta)\cdot\max\{\mu, \mu^2\}}}{\varepsilon n}\right)\right).$$

Proof. Consider the following regret function: $R_D(\mathcal{A}_{\textsf{FTRL-LS}}; \theta^*) = \frac{1}{n}\sum_{t=1}^{n}\left((y_t - \langle\theta_t, \mathbf{x}_t\rangle)^2 - (y_t - \langle\theta, \mathbf{x}_t\rangle)^2\right)$. We will bound $\mathbb{E}_D\left[R_D(\mathcal{A}_{\textsf{FTRL-LS}}; \theta^*)\right]$. Following the notation in the proof of Theorem 4.1, recall the following two functions.

- $\theta_{t+1} \leftarrow \arg\min_{\theta\in\mathcal{C}} \underbrace{\sum_{i=1}^{t}\left(\theta^\top\mathbf{x}_i\mathbf{x}_i^\top\theta - 2y_i\langle\mathbf{x}_i, \theta\rangle\right) + \frac{\lambda}{2}\|\theta\|_2^2 + \langle\boldsymbol{b}_t, \theta\rangle + \theta^\top B_t\theta}_{J_t^{\textsf{priv}}(\theta)}$, where the noise $\boldsymbol{b}_t = \sum_{i=1}^{t} y_i\mathbf{x}_i - \boldsymbol{s}_t$ with $\boldsymbol{s}_t$ being the output of $\texttt{GetSumBias}\left(\mathcal{T}_{\textsf{bias}}, t\right)$, and the noise $B_t = W_t - \sum_{i=1}^{t}\mathbf{x}_i\mathbf{x}_i^\top$ with W_t being the output of $\texttt{GetSumCov}\left(\mathcal{T}_{\textsf{cov}}, t\right)$. By standard bound on Gaussian random variables, w.p. at least $1-\beta$, $\forall t \in [n]$, $\|\boldsymbol{b}_t\|_2 = O\left(L\sigma\sqrt{p\ln(n)\ln(n/\beta)}\right)$ and $\|B_t\|_2 = O\left(L^2\sigma\sqrt{p\ln(n)\ln(n/\beta)}\right)$. We will use this bound to control the error introduced due to privacy.
- $\widetilde{\theta}_{t+1} \leftarrow \arg\min_{\theta\in\mathcal{C}} \underbrace{\sum_{i=1}^{t}\left(\theta^\top\mathbf{x}_i\mathbf{x}_i^\top\theta - 2y_i\langle\mathbf{x}_i, \theta\rangle\right) + \frac{\lambda}{2}\|\theta\|_2^2}_{J_t^{\textsf{np}}(\theta)}$.

By an analogous argument to (5) in the proof of Theorem 4.1, we have

$$\left\|\widetilde{\theta}_{t+1} - \theta_{t+1}\right\|_2 = O\left(\frac{L\sigma + L^2\sigma\cdot\mu}{\lambda}\cdot\sqrt{p\ln(n)\ln(n/\beta)}\right). \tag{8}$$

Therefore,

$$J_t^{\textsf{np}}(\widetilde{\theta}_{t+1}) + \langle\boldsymbol{b}_t, \widetilde{\theta}_{t+1}\rangle + \widetilde{\theta}_{t+1}^\top B_t\widetilde{\theta}_{t+1} \ge \underbrace{J_t^{\textsf{np}}(\theta_{t+1}) + \langle\boldsymbol{b}_t, \theta_{t+1}\rangle + \theta_{t+1}^\top B_t\theta_{t+1}}_{J_t^{\textsf{priv}}(\theta_{t+1})} + \frac{\lambda}{2}\left\|\widetilde{\theta}_{t+1} - \theta_{t+1}\right\|_2^2 \tag{9}$$

$$\Rightarrow \quad J_t^{\textsf{np}}(\theta_{t+1}) - J_t^{\textsf{np}}(\widetilde{\theta}_{t+1}) = O\left(\|\boldsymbol{b}_t\|_2\cdot\left\|\widetilde{\theta}_{t+1} - \theta_{t+1}\right\|_2 + \|B_t\|_2\cdot\left\|\widetilde{\theta}_{t+1} - \theta_{t+1}\right\|_2\cdot\mu\right) \tag{10}$$

$$\Rightarrow \quad J_t^{\textsf{np}}(\theta_{t+1}) - J_t^{\textsf{np}}(\widetilde{\theta}_{t+1}) = O\left((p\ln(n)\ln(n/\beta)\sigma^2)\cdot\frac{L^2 + L^4\mu^2 + L^3\mu}{\lambda}\right). \tag{11}$$

(9) follows from the strong convexity of J_t^{priv} and the fact that that θ_{t+1} is the minimizer of J_t^{priv}. (10) follows from the bounds on $\|\boldsymbol{b}_t\|_2$, $\|B_t\|_2$, and (8). We now use Theorem 2 from [61] to bound $\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[(y-\langle\mathbf{x},\theta_{t+1}\rangle)^2+\frac{\lambda}{2}\|\theta_{t+1}\|_2^2\right]-\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[(y-\langle\mathbf{x},\theta^*\rangle)^2+\frac{\lambda}{2}\|\theta^*\|_2^2\right]$ for any $\theta^*\in\mathcal{C}$.

Using Theorem 2 from [61] and (11), we have that w.p. at least $1-\beta$ over the randomness of the algorithm,

$$\begin{aligned}
&\mathbb{E}_{(\mathbf{x},y)\sim\tau}\left[(y-\langle\mathbf{x},\theta_{t+1}\rangle)^2+\frac{\lambda}{2}\|\theta_{t+1}\|_2^2\right]-\mathbb{E}_{(\mathbf{x},y)\sim\tau}\left[(y-\langle\mathbf{x},\theta^*\rangle)^2+\frac{\lambda}{2}\|\theta^*\|_2^2\right]\\
&\leq\frac{2}{t}\cdot\mathbb{E}\left[J_t^{\mathsf{np}}\left(\theta_{t+1}\right)-J_t^{\mathsf{np}}\left(\widetilde{\theta}_{t+1}\right)\right]+O\left(\frac{L^4\mu^2}{\lambda}\right)\\
&=O\left((p\ln(n)\ln(n/\beta))\sigma^2)\cdot\frac{L^2+L^4\mu^2+L^3\mu}{\lambda t}+\frac{L^4\mu^2}{\lambda}\right). \qquad (12)
\end{aligned}$$

(12) immediately implies the following:

$$\mathbb{E}_D\left[R_D(\mathcal{A}_{\mathsf{FTRL\text{-}LS}};\theta^*)\right]=O\left(\left(p\ln^2(n)\ln(n/\beta)\sigma^2\right)\frac{L^2+L^4\mu^2+L^3\mu}{\lambda n}+\frac{L^4\mu^2}{\lambda}+\frac{\lambda\ln(n)}{n}\cdot\|\theta^*\|_2^2\right). \qquad (13)$$

We get the regret guarantee in Theorem C.4 by optimizing for λ. □

C.3 Formal Statement of Online-to-batch Conversion for Excess Population Risk

Theorem C.5 (Corollary to Theorem 4.1 and [61])**.** *Recall the setting of parameters from Theorem 4.1, and let $\theta^{priv}=\frac{1}{n}\sum_{t=1}^{n}\theta_t$ (where $[\theta_1,\ldots,\theta_n]$ are outputs of Algorithm $\mathcal{A}_{\mathsf{FTRL}}$ (Algorithm 1). If the data set D is drawn i.i.d. from the distribution τ, then we have that w.p. at least $1-\beta$ (over the randomness of the algorithm $\mathcal{A}_{\mathsf{FTRL}}$),*

$$\mathbb{E}_D\left[\mathsf{PopRisk}(\theta^{priv})\right]=L\mu\cdot O\left(\sqrt{\frac{\ln(1/\beta)}{n}}+\sqrt{\frac{p^{1/2}\ln^2(1/\delta)\ln(1/\beta)}{\varepsilon n}}\right).$$

Here, $\mu=\max_{\theta\in\mathcal{C}}\|\theta\|_2$ is an upper bound on the norm of any model in $\mathcal{C}$.

D Multi-pass and Mini-batch DP-FTRL

We introduce two extensions to Algorithm $\mathcal{A}_{\mathsf{FTRL}}$ (Algorithm 1) that we will use for our empirical evaluation: i) Multi-pass, and ii) Mini-batching. While DP-FTRL (Algorithm 1) is stated for a single epoch of training, i.e., where each sample in the data set is used once for obtaining a gradient update, there can be situations where $E>1$ epochs of training are preferred. There are two natural ways that Algorithm 1 can be extended to the following.

DP-FTRL with Tree Restart (DP-FTRL-TR): Restarting the tree at every epoch of training. Since this amounts to adaptive composition of Algorithm 2 for E times, the privacy guarantee for this method can be obtained from Theorem 3.1 and the adaptive sequential composition property of RDP [51].

Theorem D.1 (Privacy guarantee for DP-FTRL-TR)**.** *If $\|\nabla_\theta\ell(\theta;d)\|_2\leq L$ for all $d\in\mathcal{D}$ and $\theta\in\mathcal{C}$, then DP-FTRL (Algorithm 1) with Tree Restart (DP-FTRL-TR) for E epochs satisfies $\left(\alpha,\frac{\alpha EL^2\lceil\lg(n)\rceil}{2\sigma^2}\right)$-RDP. Correspondingly, by setting $\sigma=\frac{\sqrt{EL^2\lceil\lg(n)\rceil\ln(1/\delta)}}{\varepsilon}$ one can satisfy (ε,δ)-differential privacy guarantee, as long as $\varepsilon\leq 2\ln(1/\delta)$.*

DP-FTRL with No Tree Restart (DP-FTRL-NTR): Build a single tree for all the E epochs of training.

Theorem D.2 (Privacy guarantee for DP-FTRL-NTR)**.** *If $\|\nabla_\theta\ell(\theta;d)\|_2\leq L$ for all $d\in\mathcal{D}$ and $\theta\in\mathcal{C}$, then DP-FTRL (Algorithm 1) with No Tree Restart (DP-FTRL-NTR) for E epochs satisfies $\left(\alpha,\frac{\alpha EL^2\lceil\lg(nE)\rceil}{2\sigma^2}\right)$-RDP. Correspondingly, by setting $\sigma=\frac{\sqrt{EL^2\lceil\lg(nE)\rceil\ln(1/\delta)}}{\varepsilon}$ one can satisfy (ε,δ)-differential privacy guarantee, as long as $\varepsilon\leq 2\ln(1/\delta)$.*

The proof of Theorem D.2 follows from that of Theorem 3.1, and the additional observation that any aggregation step can involve at most E gradients from any data sample $d \in \mathcal{D}$, which results in $\left\|\sum_{e\in[E]} \nabla_\theta \ell_e(\theta; d)\right\|_2 \leq EL$ from the triangle inequality. Another difference in the proof is that any data sample $d \in D$ now can affect $\lceil \lg(nE) \rceil$ nodes of the tree $\mathcal{T}$.

Mini-batch DP-FTRL: So far, for simplicity we have focused on DP-FTRL with model updates corresponding to new gradient from a single sample. However, in practice, instead of computing the gradient on a single data sample d_t at time step t, we will estimate gradient over a batch $M_t = \left\{d_t^{(1)}, \ldots, d_t^{(k)}\right\}$ as $\nabla_t = \frac{1}{k}\sum_{i=1}^{k} \mathsf{clip}\left(\nabla_\theta \ell\left(\theta_t; d_t^{(1)}\right), L\right)$. This immediately implies the number of steps per epoch to be $\lceil n/k \rceil$. Furthermore, since the ℓ_2-sensitivity in each batch gets scaled down to $\frac{L}{k}$ instead of L (as in Algorithm $\mathcal{A}_{\mathsf{FTRL}}$). W will take the above two observations into consideration in our privacy accounting accordingly.

E Omitted Details for Experiment Setup (Section 5.1)

E.1 Additional Details on Model Architectures

Table 2a shows the model architecture for MNIST and EMNIST, Table 2b shows that for CIFAR-10, and Table 2c shows the neural networks adopted from [58].

Table 2: Model architectures for all experiments.

(a) Model architecture for MNIST and EMNIST.

Layer	Parameters
Convolution	16 filters of 8×8, strides 2
Convolution	32 filters of 4×4, strides 2
Fully connected	32 units
Softmax	-

(b) Model architecture for CIFAR-10.

Layer	Parameters
Convolution $\times 2$	32 filters of 3×3, strides 1
Max-Pooling	2×2, stride 2
Convolution $\times 2$	64 filters of 3×3, strides 1
Max-Pooling	2×2, stride 2
Convolution $\times 2$	128 filters of 3×3, strides 1
Max-Pooling	2×2, stride 2
Fully connected	128 units
Softmax	-

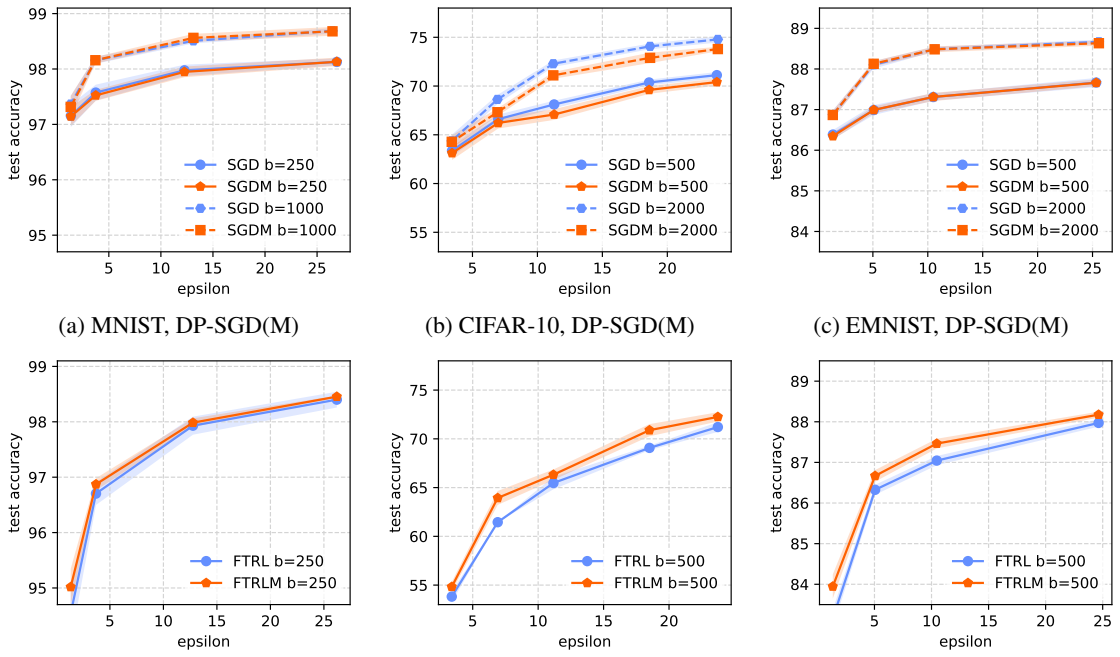
(c) Model architecture for StackOverflow. [58]

Layer	Output Shape	Parameters
Input	20	0
Embedding	(20, 96)	960384
LSTM	(20,670)	2055560
Dense	(20, 96)	64416
Dense	(20, 10004)	970388
Softmax	-	-

E.2 Comparison of Optimizers with their Momentum Variants

Figure 3 shows an comparison between the original and the momentum versions of DP-FTRL and DP-SGD on the three centralized example-level DP image classification tasks. We can see that for any privacy level, the utility of DP-SGD is at least that of DP-SGDM (sometimes even more). Moreover, we see that DP-FTRLM always outperforms DP-FTRL.

The experiments in Table 3 and Figure 4 show the advantages of the momentum variant for the federated StackOverflow task in practice. We compare DP-SGD and its momentum variant DP-SGDM, DP-FTRL and its momentum



(a) MNIST, DP-SGD(M) (b) CIFAR-10, DP-SGD(M) (c) EMNIST, DP-SGD(M)

(d) MNIST, DP-FTRL(M) (e) CIFAR-10, DP-FTRL(M) (f) EMNIST, DP-FTRL(M)

Figure 3: Final test accuracy vs. privacy (example-level ε) for various noise multipliers. DP-FTRLM outperforms DP-FTRL, and DP-SGD is always not worse than DP-SGDM.

variant DP-FTRLM under two different privacy epsilons. Privacy epsilon is infinite when noise multiplier is zero; privacy epsilon is 8.53 when noise multiplier is 0.4 for DP-SGD and DP-SGDM; privacy epsilon is 8.5 when noise multiplier is 2.33 for DP-FTRL and DP-FTRLM. We tune and select the hyperparameter with the best validation accuracy [8]. We then run the experiment with the specific set of hyperparameters for five times to estimate mean and standard deviation of the accuracy.

The momentum variant helps in two ways for StackOverflow: momentum significantly improve the performance of both SGD and FTRL when the noise is relatively small; moreover, momentum stabilizes DP-FTRL when the noise is relatively large. Note that the tree aggregation method in DP-FTRL use different privacy calculation method compared to DP-SGD. A relatively large noise multiplier has to be used to achieve the same privacy ε guarantee. While tree aggregation in DP-FTRL exploits the $O(\log n)$ accumulated noise, it also introduces unstable jump for the noise added in each round, which could be mitigated by the momentum γ introduced in DP-FTRLM. In the experiments of StackOverflow, we will always use the momentum variant unless otherwise specified.

F Omitted Details for Experiments in Section 5.2

F.1 Details of Hyperparameter Tuning

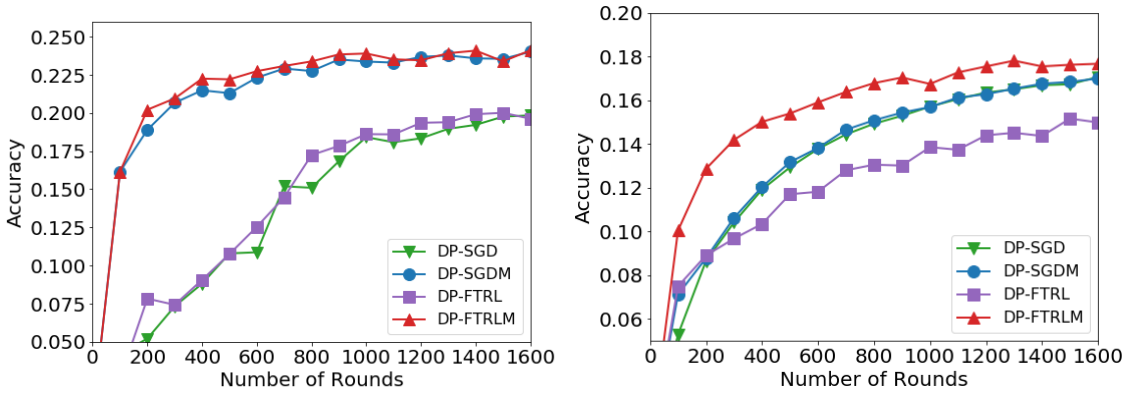
Image classification experiments For the three image classification experiments, we tune the learning rate ($1/\lambda$ for FTRL) over a grid of the form $\cup_{i\in\{-3,-2,\ldots,3\}}\{10^i, 2\times 10^i, 5\times 10^i\}$, selecting the value that achieves the highest test accuracy averaged over the last 5 epochs while ensuring this chosen value is not an endpoint of the grid. We use a clipping norm 1.0 for all the image classification experiments following previous work [54].

The parameter search for non-private baseline is the same as that for the DP algorithms. We use regular SGD (with

[8] The accuracy for StackOverflow next word prediction task excludes the end of sequence symbol and the out of vocabulary symbol following [58]. The hyperparameters tuning range are described in Appendix F.1.

Server Optimizer	Epsilon	Accuracy		Hyperparameters		
		Validation	Test	ServerLR	ClientLR	Clip
DP-SGD	∞	$19.62 \pm .12$	$20.99 \pm .11$	3	0.5	1
DP-SGDM		$23.87 \pm .22$	$24.89 \pm .27$	3	0.5	1
DP-FTRL		$19.95 \pm .05$	$21.12 \pm .14$	3	0.5	1
DP-FTRLM		$23.89 \pm .03$	$25.15 \pm .07$	3	0.5	1
DP-SGD	8.53	$16.83 \pm .05$	$18.25 \pm .05$	3	0.5	0.3
DP-SGDM		$16.92 \pm .03$	$18.27 \pm .04$	0.1	0.5	1
DP-FTRL	8.50	$15.04 \pm .16$	$15.46 \pm .39$	3	0.5	0.3
DP-FTRLM		$17.78 \pm .08$	$18.86 \pm .15$	1	0.5	0.3

Table 3: Validation and test accuracy for the StackOverflow next word prediction task. Each experiment is run five times to calculate the mean and standard deviation. The momentum variant DP-FTRLM performs better than DP-FTRL.



(a) Privacy epsilon ∞ (b) Privacy epsilon ~ 8.5

Figure 4: Training curves show validation accuracy of StackOverflow. The curve of the best validation accuracy out of the five runs is presented. The momentum variant converges faster and performs better.

and without momentum) for the image classification tasks. The chosen hyperparameters and privacy parameters are summarized in Table 6.

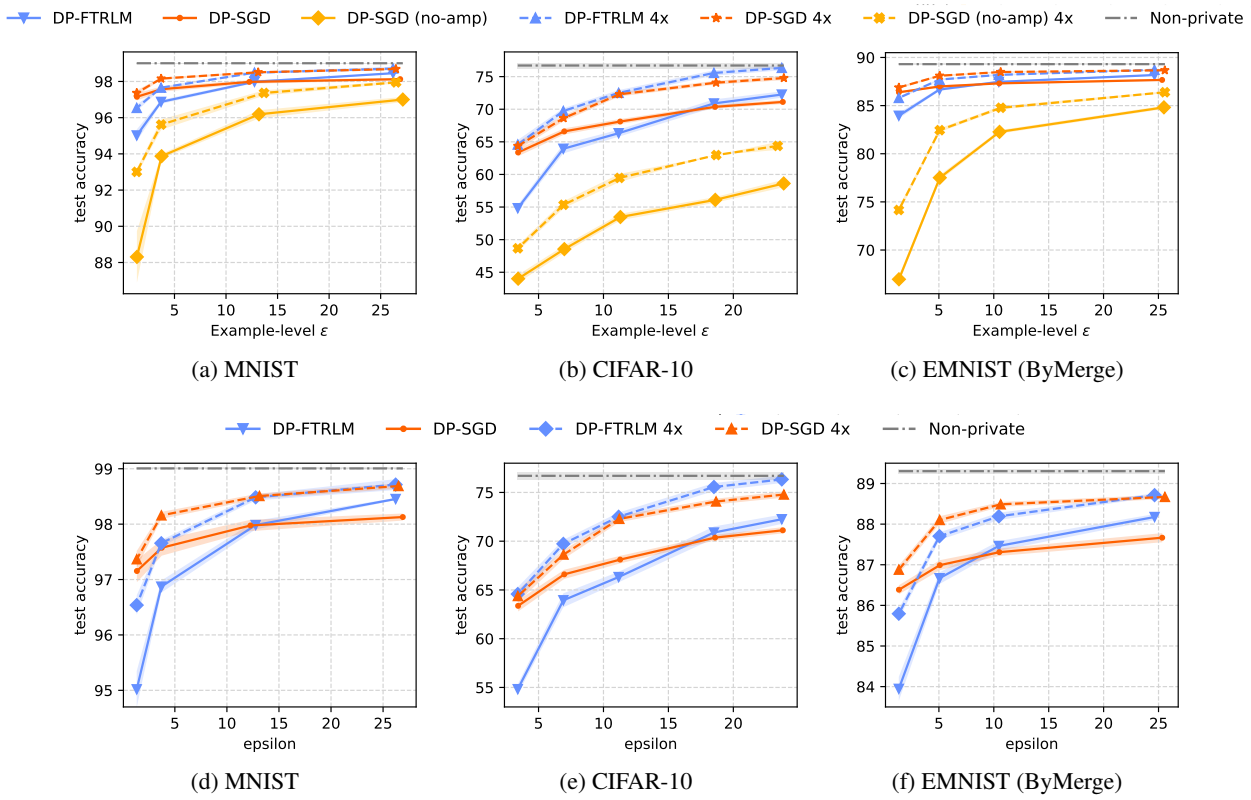
StackOverflow experiments The StackOverflow benchmark dataset of the next word prediction task has 342,477 users (clients) with training 135,818,730 examples. A validation set of 10,000 examples, and a test set of 16,576,035 examples are constructed following [58]. The one layer LSTM described in [58] is used. We compare with DP-FedAvg where DP-SGD is used on server.

There are many hyperparameters in federated learning. We fix the number of total rounds to be 1,600 for StackOverflow, and sample 100 clients per round for DP-SGD, and take 100 clients from the shuffled clients for DP-FTRL to make sure the clients are disjoint across rounds. Note that DP-FTRL would run less than one epoch for StackOverflow. On each client, the number of local epochs is fixed to be one and the batch size is sixteen, and we constrained the maximum number of samples on each client to be 256. The momentum for both DP-SGDM and DP-FTRLM is fixed to 0.9.

In most of the experiments, we will tune server learning rate, client learning rate and clip norm for a certain noise multiplier. We tune a relative large grid (client learning rate in $\{0.1, 0.2, 0.5, 1, 2\}$, server learning rate in $\{0.03, 0.1, 0.3, 1, 3\}$, clip norm in $\{0.1, 0.3, 1, 3, 10\}$) when the noise multiplier is zero. And we have several observation: the best accuracy of clip norm 0.3 and 1.0 are slightly better than larger clip norms, which suggests that clip norm could generally help for this language task; increasing server learning rate could complement decreasing clip

norm when clip norm is effective; the largest client learning rate that does not diverge often leads to good final accuracy. As adding noise increases the variance of gradients, we often have to decrease learning rate in practice. Based on this heuristic and the observation from tuning when noise multiplier is zero, we choose client learning rate from $\{0.1, 0.2, 0.5\}$, server learning rate from $\{0.1, 0.3, 1, 3\}$ and clip norm from $\{0.3, 1, 3\}$ unless otherwise specified. We use DP-SGD with zero noise for StackOverflow, as gradient clipping can improves accuracy for language tasks.

F.2 Omitted Details for Image Classification Experiments



(a) MNIST (b) CIFAR-10 (c) EMNIST (ByMerge)

(d) MNIST (e) CIFAR-10 (f) EMNIST (ByMerge)

Figure 5: Final test accuracy vs. ε. The lines show the mean and standard deviation over 5 runs. Top row shows the full plot of Figure 1. Bottom row shows only the comparison between DP-FTRL and DP-SGD with amplification.

In Figure 5, we plot a comparison between DP-FTRL and DP-SGD with amplification for two batch sizes, i.e., in addition to the curves in Figure 1, we plot the DP-SGD with amplification at the higher batch size. We can see that for both batch sizes, DP-SGD with amplification outperform DP-FTRLM at small ε, while DP-FTRLM outperforms DP-SGD when ε increases.

One might notice that the crossover point at which DP-FTRLM starts to outperform DP-SGD changes with batch size, and one might wonder if the point should shift to the left or right as batch size increases. We can see the crossover point shifts to the left for CIFAR-10, shifts to the right for EMNIST, and remains roughly the same for MNIST. We conjecture that the direction of shift would highly depends on the batch size and the number of training examples, which affect the privacy amplification analysis. When the ratio between the batch size and the training set size is small, we would likely see a shifting towards the right; and when the ratio is larger, we would likely observe a left shifting. This can be backed up by Figure 8, where we can see the two ε-batch curves for a specific accuracy α crossing at two points. Denote the crossing points as (b_1, ε_1) and (b_2, ε_2). We know that for batch size b_1 (or b_2), if we plot the accuracy-ε curves for DP-SGD and DP-FTRL as in Figure 1, we would see the crossover points at ε_1 (or ε_2).

Now we consider batch size $b_3 \gtrapprox b_1$ that corresponds to privacy levels ε_3^S for DP-SGD and ε_3^F for DP-FTRL. From

the shape of the curves, we have $\varepsilon_3^S < \varepsilon_3^F \approx \varepsilon_1$. Considering the accuracy-ε curve for b_3, we know that DP-SGD has reached accuracy α at ε_3^S while DP-FTRL only reaches α at a larger privacy level ε_3^F. Therefore, we know that at $\varepsilon_3^F \approx \varepsilon_2$, DP-SGD still reaches higher accuracy than DP-FTRL, i.e., the crossover has not yet happen at this privacy level. Therefore, when we increase batch size from b_1 to b_3, we would likely see the crossover point shifting toward the right.

Then we consider batch size $b_4 \gtrapprox b_2$ that corresponds to privacy levels ε_4^S for DP-SGD and ε_4^F for DP-FTRL. As the DP-SGD curve is pretty flat in this regime, we have $\varepsilon_2 > \varepsilon_4^S > \varepsilon_4^F$. Considering the accuracy-ε curve for b_4, we know that DP-FTRL has reached accuracy α at ε_4^F while DP-SGD only reaches α at a larger privacy level ε_4^S. Therefore, we know that at ε_4^F, DP-FTRL have already reached a higher accuracy than DP-SGD, i.e., the crossover has already happened before ε_4^F. Therefore, when we increase batch size from b_2 to b_4, we would see the crossover point shifting toward the left.

F.3 Omitted Details for StackOverflow Experiments

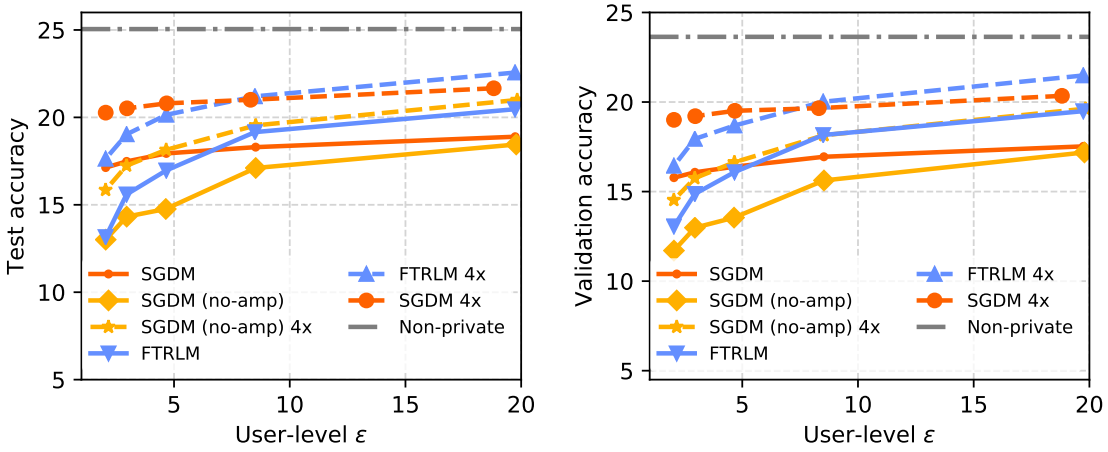


Figure 6: Test and Validation accuracy for the StackOverflow next word prediction task under different privacy epsilon by varying noise multipliers.

Server Optimizer	Epsilon	Accuracy		Hyperparameters			
		Validation	Test	Noise	ServerLR	ClientLR	Clip
DP-SGDM	19.74	17.52	18.89	0.3	1	0.5	0.3
DP-FTRLM	19.74	19.49	20.47	1.13	0.3	0.5	1
DP-SGDM	8.53	16.94	18.30	0.4	0.1	0.5	1
DP-FTRLM	8.50	18.16	19.16	2.33	1	0.5	0.3
DP-SGDM	4.66	16.39	17.94	0.5	0.3	0.5	0.3
DP-FTRLM	4.66	16.09	16.97	4.03	0.1	0.5	1
DP-SGDM	2.95	16.08	17.48	0.6	0.3	0.5	0.3
DP-FTRLM	2.95	14.87	15.60	6.21	0.3	0.5	0.3
DP-SGDM	2.05	15.78	17.13	0.7	0.3	0.5	0.3
DP-FTRLM	2.04	13.06	13.16	8.83	0.3	0.2	0.3

Table 4: Validation and test accuracy for the StackOverflow next word prediction task under different privacy epsilon.

We compare the accuracy of the momentum variant of DP-FTRL with the momentum variant of DP-SGD as

baseline under different privacy epsilon. We tune hyperparameters as described in Appendix F.1 and select the hyperparameters achieve the best validation accuracy for StackOverflow (see Table 4 and Figure 6). DP-FTRLM performs better than DP-SGDM when the epsilon is relatively large, but performs worse when the epsilon is small. More noise are added to DP-FTRLM to achieve the same privacy epsilon as DP-SGDM. However, DP-FTRLM can result in utility (accuracy) not (much) worse than DP-SGDM without relying on amplification by sampling, which makes it appealing for practical federated learning setting where population and sampling is difficult to estimate [5]. Note that the noise added for both DP-FTRLM and DP-SGDM are considered large for federated learning tasks. The effective noise could be significantly reduced by sampling more clients each round in practice [49], and more discussion on this front is in Appendix G.

G Omitted Details for Experiments in Section 5.3

G.1 Effect of batch size for privacy/computation trade-offs

We set a target utility level based on what might be achieved at large ε in Figure 1, and examine if increasing batch size can lead to better privacy-utility trade-offs.

First, for all three datasets, Figure 7 shows the accuracy trajectories of three different batch sizes with scaled noise, i.e., for batch sizes b_1, b_2, b_3 and noise σ_1, σ_2, σ_3, we have $\sigma_1/b_1 = \sigma_2/b_2 = \sigma_3/b_3$. We can observe that for both FTRL with momentum and DP-SGD, the training trajectories for noise and batch size pairs (b_1, σ_1), (b_2, σ_2), and (b_3, σ_3) are roughly the same. Notice that different (b_i, σ_i) leads to different ε values. The hyperparameters and privacy parameters in the experiments can be found in Table 6.

As we have confirmed that scaling the batch size and noise together does not affect the accuracy, in Figure 8, we plot the ε value versus batch size b such that σ/b is a fixed value. We can see that as batch size grows, FTRL achieves better privacy than DP-SGD at the same level of accuracy.

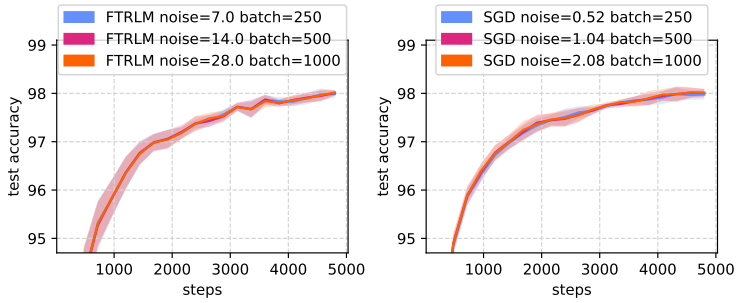
G.2 Details of Hyperparameter Tuning

In Appendix F.3, a significant amount of noise has to be added in both DP-FTRLM and DP-SGDM to achieve nontrivial privacy epsilons, which leads to undesired accuracy degradation. For example, the test accuracy of DP-FTRLM on StackOverflow dataset decreases from 25.15% when $\varepsilon = \infty$ to 18.86% when $\varepsilon = 8.5$ when the number of clients per round is fixed at 100. In practical federated learning tasks, the total population is very large and many more clients could be sampled every round. In this section, taking StackOverflow as an example, we study the minimum number of sampled clients per round (report goal in [11]) to achieve a target accuracy under certain privacy budget.

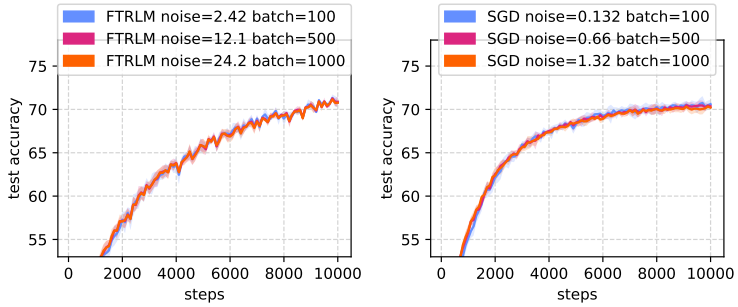
Fix the clip norm and client learning rate to reduce hyperparameter tuning complexity. We first find the largest noise multiplier that would meet the target accuracy based on selecting 100 clients per round. As an extensive grid search over noise multiplier while simultaneously tuning server learning rate, client learning rate and clip norm is computationally intensive, we fix the clip norm to 1 and the client learning rate to 0.5 based on Figure 9. We then tune the server learning rate from $\{0.3, 1, 3\}$ for each noise multiplier.

Grid search for the largest noise multiplier to meet the target. We use a grid of ten noise multipliers between 0 ($\varepsilon = \infty$, test accuracy=24.89) and 0.3 ($\varepsilon = 18.89$, test accuracy=18.89) for DP-SGDM, and between 0 ($\varepsilon = \infty$, test accuracy=25.15) and 1.13 ($\varepsilon = 19.74$, test accuracy=20.47) for DP-FTRLM. And we further add five noise multipliers between 0 and 0.035 for DP-SGDM, and between 0 and 0.149 for DP-FTRLM based on the results of the previous grid search on ten noise multipliers. The test accuracy is presented in Figure 2b. We set the target test accuracy as 24.5% and select noise multiplier 0.007 (with server learning rate 3) for DP-SGDM and noise multiplier 0.067 (with server learning rate 3) for DP-FTRLM.

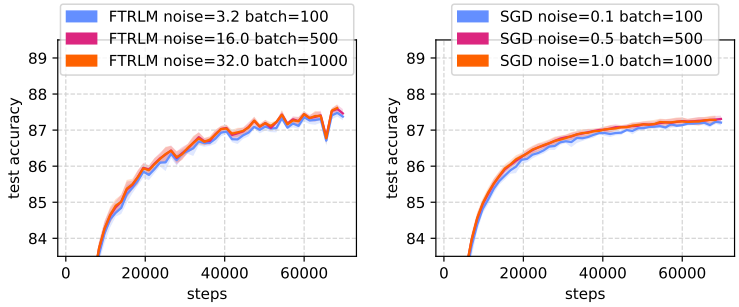
Report goal for the nontrivial privacy epsilon in practice. The standard deviation of noise added in each round is proportional to the inverse of the number of clients per round (report goal). The practical federated learning tasks



(a) MNIST.

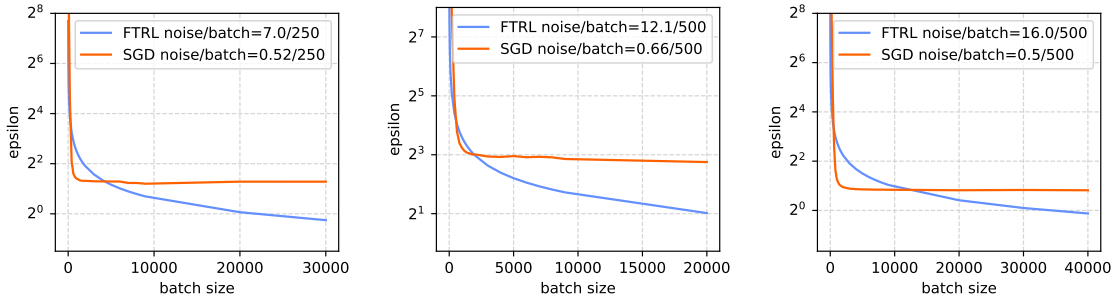


(b) CIFAR-10.



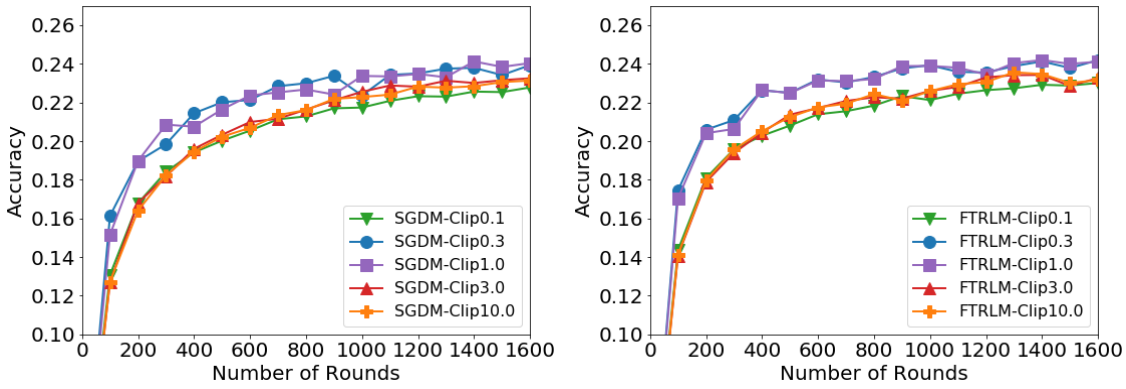
(c) EMNIST (ByMerge).

Figure 7: Batch size with noise scaled proportionally (so the amount of noise in the average gradient remains constant) does not affect accuracy. Thus, we can use a single run with a given noise level σ and batch size b to estimate the accuracy we would get with noise level $\alpha\sigma$ and batch size αb for small α.



(a) MNIST. (b) CIFAR-10. (c) EMNIST (ByMerge).

Figure 8: ε vs. batch size. According to Figure 7, DP-FTRL and DP-SGD with the corresponding noises achieve roughly the same accuracy.



(a) SGDM with zero noise

(b) FTRLM with zero noise

Figure 9: Training curve of the best validation accuracy under various clip norm for StackOverflow.

often have a very large population and report goal, and we could simultaneously increase the noise multiplier and report goal, so that the utility (accuracy for classification and prediction tasks) will likely not degrade [49] while the privacy guarantee is improved. The validation accuracy of simulation performance with two different report goals for StackOverflow is presented in Figure 10. The noise multiplier 0.067 is used for DP-FTRLM and 0.007 is used for DP-SGD when report goal is 100, which is the largest noise multiplier to meet the target test accuracy determined by Figure 2b. We run each experiment for five times and plot the curves for the median validation accuracy, the corresponding test accuracy are 24.73% for DP-SGDM and 24.63% for DP-FTRLM. We then run the same experiments with report goal of 1000, and proportionally increase the corresponding noise multiplier to be 0.67 for DP-FTRLM and 0.07 for DP-SGDM. The performance of 1000 report goal is slightly better with test accuracy 25.19% for DP-SGDM and 24.97% for DP-FTRLM. We will assume the utility will not decrease if report goal and noise multiplier are simultaneously and proportionally increased.

As shown in Table 5, both report goals 100 and 1000 would provide trivial privacy guarantee of large epsilon for the target utility. We have to increase the report goal to $2.06e4$ to get a nontrivial privacy epsilon (less than 10) with DP-FTRLM and the StackOverflow population of $3.42e5$ [9]. Smaller report goal could achieve similar privacy guarantee if the population becomes larger. In Figure 2c, the relationship between privacy guarantee and report goal for DP-FTRLM and DP-SGDM are presented. DP-FTRLM provides better privacy guarantee by smaller report goal when the privacy epsilon is relatively large or very small. The range where DP-FTRLM outperforms DP-SGDM in report goals and privacy guarantees are larger when the population is relatively small or very large.

G.3 Increasing population size for privacy/computation trade-off

Though the plots in Figure 2c use the actual population size of 340k in StackOverflow for their privacy computation, in Figure 11 we show a similar plot for a *hypothetical* population size of 1M clients. It is easy to see that the privacy-computational cost trade-off for both the techniques improves[10], more so for DP-SGD since the amplification improves due to lower sampling rate. However, it is still the case that DP-FTRLM provides a better trade-off than DP-SGDM for privacy parameter $\varepsilon \notin [3.2, 10]$ at $\delta = 10^{-6}$ for utility target 24.5%, and nearly all $\varepsilon \in (0, 50]$ for utility target 23%.

[9] The best epsilon DP-SGDM can achieve is 10.16 by increasing report goal to be as large as the population $3.42e5$

[10] The "kink" at $\varepsilon \approx 15$ in the curve labelled "DP-FTRLM 24.5%" is due to the fact that the privacy accounting in DP-FTRL depends on the maximum number of times any client participates in training. For report goal 500, all clients need to participate at most once, whereas for report goal 700, the required number of rounds are just enough for a few clients to need to participate twice, which accounts for the visibly increased privacy cost. In fact, for a higher report goal of 1000, still no client needs to participate more than twice to complete 1600 training rounds.

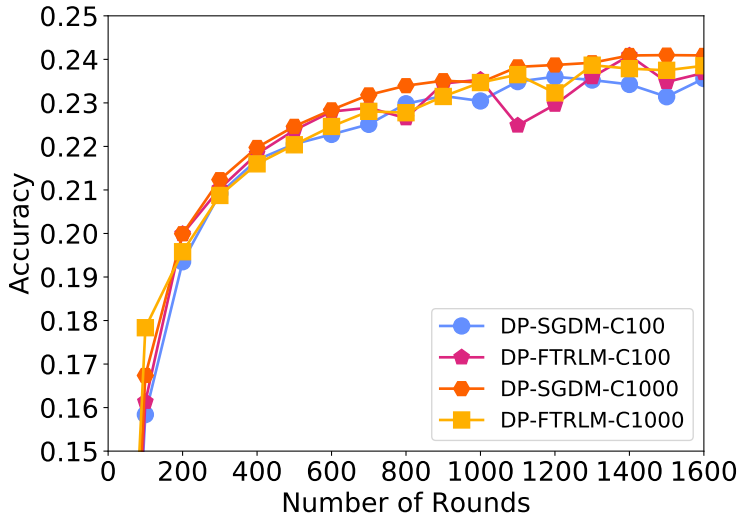


Figure 10: Training curves of validation accuracy for DP-SGDM and DP-FTRLM for StackOverflow for report goal 100 and 1000 (suffix C100 and C1000 in the legend). DP-FTRLM with restart (see Appendix D) is used when report goal is 1000 (less than five epochs of data). Simultaneously increasing noise multiplier and report goal by 10x could significantly improve the privacy guarantee without sacrificing the utility. The noise multiplier for DP-SGDM-C100, DP-FTRLM-C100, DP-SGDM-C1000, DP-FTRLM-C1000 are 0.007, 0.067, 0.07, and 0.67, respectively. The corresponding test accuracy are 24.73%, 24.63%, 25.19% and 24.93%. The corresponding privacy ε can be found in Table 5 .

Server Optimizer	Privacy		Setting		
	Epsilon	Delta	Noise	Report goal	Population
DP-SGDM	1.78e7	1e-6	0.007	100	3.42e5
DP-FTRLM	1.49e3	1e-6	0.067	100	3.42e5
DP-SGDM	7.71e4	1e-6	0.07	1000	3.42e5
DP-FTRLM	1.03e2	1e-6	0.67	1000	3.42e5
DP-SGDM	10.16	1e-6	23.97	3.42e5	3.42e5
DP-FTRLM	9.66	1e-6	13.81	2.06e4	3.42e5
DP-FTRLM	4.35	1e-6	35.24	5.26e4	3.42e5
DP-SGDM	8.98	1e-6	.67	9.56e3	1e6
DP-FTRLM	8.99	1e-6	8.71	1.15e4	1e6
DP-SGDM	4.17	1e-6	1.20	1.71e4	1e6
DP-FTRLM	4.19	1e-6	21.64	3.23e4	1e6

Table 5: The (ε, δ) privacy guarantee for DP-FTRLM and DP-SGDM under realistic and hypothetical report goal and population of StackOverflow that would meet the target test accuracy 24.5%. Note that the DP-FTRLM privacy accounting is based on the restart strategy in Appendix D.

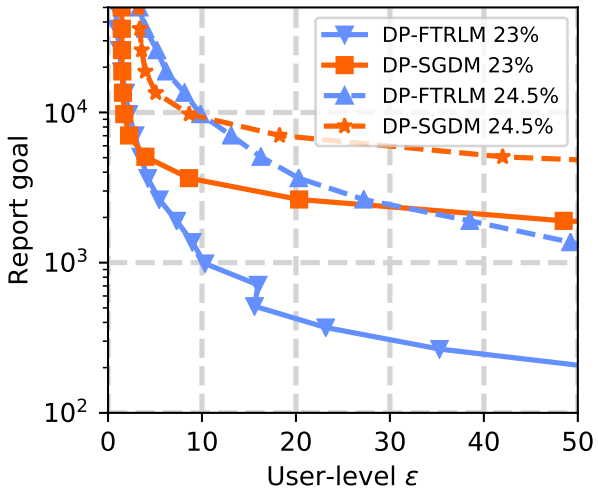


Figure 11: Relationship between privacy ε (when $\delta = 1/\text{population}$) and report goal for a fixed accuracy target for DP-FTRLM and DP-SGDM on the StackOverflow dataset with a hypothetically larger population of 1M users.

Table 6: Parameters for the image classification experiment in Figure 1 (and the full version Figure 5) and Figure 7. Clipping norm is 1.0. The "learning rate" reported for FTRL(M) is λ.

(a) MNIST

$b = 250$ 20 epochs	FTRL/FTRLM	noise	4.0	7.0	20.0	50.0
		ε	26.21	12.76	3.70	1.34
		learning rate	1.0/10.0	2.0/20.0	5.0/50.0	10.0/100.0
	SGD/SGDM	noise	0.42	0.52	0.74	1.14
		ε	26.90	12.26	3.75	1.35
		learning rate	0.5/0.05	0.5/0.05	0.5/0.05	0.2/0.02
	SGD unamplified	noise	1.08	1.89	5.48	13.7
		ε	27.15	13.19	3.75	1.35
		learning rate	0.2	0.2	0.05	0.02
$b = 1000$ 80 epochs	FTRLM	noise	8.0	14.0	40.0	100.0
		ε	26.21	12.76	3.70	1.34
		learning rate	5.0	10.0	20.0	50.0
	SGD/SGDM	noise	0.62	0.8	1.61	3.67
		ε	26.48	13.12	3.71	1.34
		learning rate	2.0/0.2	2.0/0.1	0.5/0.05	0.2/0.02
	SGD unamplified	noise	2.2	3.67	11.06	27.68
		ε	26.50	13.68	3.71	1.34
		learning rate	0.5	0.2	0.1	0.05

(b) CIFAR-10

$b = 500$ 100 epochs	FTRL/FTRLM	noise	10	12.1	18.1	27	50
		ε	23.73	18.51	11.17	6.91	3.40
		learning rate	2.0/20.0	2.0/20.0	5.0/50.0	5.0/50.0	20.0/200.0
	SGD/SGDM	noise	0.61	0.66	0.79	0.98	1.51
		ε	23.81	18.60	11.29	6.98	3.43
		learning rate	0.5/0.05	0.5/0.05	0.5/0.05	0.2/0.02	0.2/0.02
	SGD unamplified	noise	2.66	3.22	4.79	7.15	13.26
		ε	23.88	18.61	11.30	6.99	3.43
		learning rate	0.1	0.1	0.05	0.02	0.02
$b = 2000$ 400 epochs	FTRLM	noise	20	24.2	36.2	54	100
		ε	23.73	18.51	11.17	6.91	3.40
		learning rate	10.0	10.0	20.0	20.0	50.0
	SGD/SGDM	noise	1.26	1.46	2.06	2.98	5.4
		ε	23.89	18.67	11.22	6.92	3.41
		learning rate	1.0/0.1	0.5/0.05	0.5/0.05	0.5/0.05	0.2/0.02
	SGD unamplified	noise	5.4	6.42	9.63	14.4	26.68
		ε	23.42	18.68	11.23	6.93	3.41
		learning rate	0.2	0.2	0.1	0.05	0.02

(c) EMNIST

$b = 500$ 50 epochs	FTRL/FTRLM	noise	8.0	16.0	30.0	100.0
		ε	24.64	10.46	5.06	1.35
		learning rate	2.0/20.0	2.0/20.0	5.0/50.0	10.0/100.0
	SGD/SGDM	noise	0.41	0.5	0.6	0.97
		ε	25.34	10.50	5.08	1.36
		learning rate	0.5/0.05	0.5/0.05	0.2/0.02	0.2/0.02
	SGD unamplified	noise	1.89	3.86	7.24	24.06
		ε	25.50	10.53	5.08	1.36
		learning rate	0.1	0.05	0.02	0.01
$b = 2000$ 200 epochs	FTRLM	noise	16.0	32.0	60.0	200.0
		ε	24.64	10.46	5.06	1.35
		learning rate	10.0	20.0	20.0	50.0
	SGD	noise	0.56	0.73	1.02	2.69
		ε	25.58	10.63	5.07	1.35
		learning rate	1.0/0.1	1.0/0.1	1.0/0.05	0.2/0.02
	SGD unamplified	noise	3.77	7.65	14.5	48.42
		ε	25.59	10.64	5.08	1.35
		learning rate	0.2	0.1	0.05	0.02

Table 7: Parameters for the image classification experiment in Figure 7. Clipping norm is 1.0. The "learning rate" reported for FTRL(M) is λ.

Data	Batch size		FTRLM	DPSGD
MNIST	500	noise	14.0	1.04
		ε	8.37	3.40
		learning rate	20.0	0.05
	1000	noise	28.0	2.08
		ε	5.57	2.64
		learning rate	20.0	0.05
CIFAR-10	100	noise	2.42	0.132
		ε	55.44	2389.37
		learning rate	20.0	0.05
	1000	noise	24.2	1.32
		ε	11.96	9.35
		learning rate	20.0	0.05
EMNIST	100	noise	3.2	0.1
		ε	28.51	10051.63
		learning rate	20.0	0.05
	1000	noise	32.0	1.0
		ε	6.97	2.44
		learning rate	20.0	0.05