A. Other Related Work

Differentially private empirical risk minimization (ERM) and private online learning are well-studied areas in the privacy literature (Chaudhuri et al., 2011; Kifer et al., 2012; Jain et al., 2012; Smith & Thakurta, 2013; Song et al., 2013; Bassily et al., 2014; Jain & Thakurta, 2014; Abadi et al., 2016; McMahan et al., 2017b; Wu et al., 2017; Agarwal & Singh, 2017; Abernethy et al., 2019; Bassily et al., 2019b; Iyengar et al., 2019; Pichapati et al., 2019; Thakkar et al., 2019; Feldman et al., 2020a; Papernot et al., 2020b)[8]. The connection between private ERM and private online learning was first explored in (Jain et al., 2012), and the idea of using stability induced by differential privacy for designing low-regret algorithms was explored in (Kalai & Vempala, 2005; Agarwal & Singh, 2017; Abernethy et al., 2019). 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 root of $\mathcal{T}$ to the t-th leaf node, with $\texttt{node}_1$ being the root node and $\texttt{node}_h$ being the leaf node.
   (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 $\boldsymbol{s}$.

Incorporating the iterative estimator from (Honaker, 2015): Here, we state a variant of the $\texttt{GetSum}\,(\mathcal{T}, t)$ function (called $\texttt{GetSumReducedVariance}\,(\mathcal{T}, t)$) based on the variance reduction technique used in (Honaker, 2015). The main idea is as follows: In the estimator for $\texttt{GetSum}\,(\mathcal{T}, t)$ above, each $\texttt{node}_j$ refers to a noisy/private estimate of all the nodes in the sub-tree of $\mathcal{T}$ rooted at $\texttt{node}_j$. Notice that one can obtain independent estimates of the same, with one for each level of the sub-tree rooted at $\texttt{node}_j$, by summing up the nodes at the corresponding level. Of course, the variance of each of these estimates will be different. (Honaker, 2015) provided an estimator to combine these independent estimates in order to lower the overall variance in the final estimate. In the following, we provide the formal description of $\texttt{GetSumReducedVariance}\,(\mathcal{T}, t)$. The text colored in blue is the only difference from $\texttt{GetSum}\,(\mathcal{T}, t)$.

3. $\texttt{GetSumReducedVariance}\,(\mathcal{T}, t)$: Let $[\texttt{node}_1, \ldots, \texttt{node}_h]$ be the list of nodes from the root of $\mathcal{T}$ to the t-th leaf node, with $\texttt{node}_1$ being the root node and $\texttt{node}_h$ being the leaf node.
   (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 do the following. Here if $\texttt{node}_j$ is the left child, then it is treated as its own left sibling.
      i. For any two node indices $\texttt{left} \le \texttt{right}$, let $r_{\texttt{left:right}} \leftarrow$ value in $\mathcal{T}$ corresponding to the least common ancestor of $\texttt{left}$ and $\texttt{right}$.
      ii. For the sub-tree rooted at $\texttt{node}_j$, let $[a : b]$ be the indices of the leaf nodes in the original tree $\mathcal{T}$.
      iii. Estimate $\boldsymbol{s}_{[x:z]}$ recursively as follows, with $y = \lfloor (x+z)/2 \rfloor$

$$\boldsymbol{s}_{[x:z]} \leftarrow \frac{\boldsymbol{r}'_{[x:z]}}{2 - (x - z + 1)^{-1}}, \quad \text{where } \boldsymbol{r}'_{[x:z]} \leftarrow \boldsymbol{r}_{[x:z]} + \frac{\boldsymbol{r}'_{[x:y]} + \boldsymbol{r}'_{[y+1:z]}}{2}.$$

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

iv. Add $\boldsymbol{s}_{[x:z]}$ to $\boldsymbol{s}$.

(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 (Dwork et al., 2006a),(Mironov, 2017, Corollary 3), and adaptive RDP composition (Mironov, 2017, Proposition1) to complete the proof.

While the original work on tree aggregation (Dwork et al., 2010; Chan et al., 2011) 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 $\theta_{t+1}^{\text{Noisy-SGD}} \leftarrow \theta_t - \eta \cdot \left(\nabla_\theta \ell \left(\theta_t^{\text{Noisy-SGD}}; d_t \right) + \boldsymbol{a}_t \right)$, where $\boldsymbol{a}_t$'s are noise random variables. Let the DP-FTRL (Algorithm 1) updates be $\theta_{t+1}^{\text{DP-FTRL}} \leftarrow \arg\min_{\theta \in \mathbb{R}^p} \sum_{i=1}^{t} \nabla_\theta \ell \left(\theta_i^{\text{DP-FTRL}}; d_i \right) + \langle \boldsymbol{b}_t, \theta \rangle + \frac{1}{2\eta} \|\theta\|_2^2$, 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 $t \in [n]$, $\theta_t^{\text{Noisy-SGD}} = \theta_t^{\text{DP-FTRL}}$.

Proof. Consider the non-private SGD and FTRL. Recall that the SGD update is $\theta_{t+1}^{\text{SGD}} \leftarrow \theta_t^{\text{SGD}} - \eta \nabla_\theta \ell(\theta_t^{\text{SGD}}; d_t)$, where η is the learning rate. Opening up the recurrence, we have $\theta_{t+1}^{\text{SGD}} \leftarrow \theta_0 - \eta \sum_{i=1}^{t} \nabla_\theta \ell(\theta_i^{\text{SGD}}; d_i)$. If $\theta_0^{\text{SGD}} = \mathbf{0}^p$, then equivalently $\theta_{t+1}^{\text{SGD}} \leftarrow \arg\min_{\theta \in \mathbb{R}^p} \langle \sum_{i=1}^{t} \nabla_\theta \ell(\theta_i^{\text{SGD}}; d_i), \theta \rangle + \frac{1}{2\eta} \|\theta\|_2^2$. 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 $\theta_{t+1}^{\text{Noisy-SGD}} \leftarrow \theta_t^{\text{Noisy-SGD}} - \eta \left(\nabla_\theta \ell(\theta_t^{\text{Noisy-SGD}}; d_t) + \boldsymbol{a}_t \right)$, where $\boldsymbol{a}_t$ is the Gaussian noise added at time step t. Similar as before, this rule can be written as

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

The update rule of DP-FTRL can be written as

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

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 $[\theta_1, \dots, \theta_n]$ 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

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

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 (McMahan, 2017) 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 (Shalev-Shwartz, 2012; Hazan, 2019), we have the following. For $\nabla_t = \nabla_\theta\ell(\theta_t; d_t)$,

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

One can bound the term A in (6) by (Hazan, 2019, Theorem 5.2) and get $A \leq \left(\frac{L^2}{\lambda} + \frac{\lambda}{2n}\left(\|\theta^*\|_2^2 - \|\theta_1\|_2^2\right)\right)$. As for term B, using (5) and the concentration on $\boldsymbol{b}_t$ mentioned earlier, we have, w.p. at least $1-\beta$,

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

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 $\mathcal{A}_{\textsf{FTRL-LS}}$: 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.

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

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

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

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

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

  $\theta_{t+1} \leftarrow \arg\min_{\theta\in\mathcal{C}} \left(\theta^\top \cdot W_t \cdot \theta - 2\langle \boldsymbol{s}_t, \theta\rangle\right) + \frac{\lambda}{2}\|\theta\|_2^2$. **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 $\|\mathbf{x}\|_2 \leq L$ and $|y| \leq 1$ for all $(\mathbf{x}, y) \in \mathcal{D}$ and $\theta \in \mathcal{C}$, then Algorithm 1 (Algorithm $\mathcal{A}_{\textsf{FTRL}}$) 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 $\varepsilon \leq 2\ln(1/\delta)$.*

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

- $\theta_{t+1} \leftarrow \arg\min_{\theta\in\mathcal{C}} \underbrace{\sum_{i=1}^{t}\left(\theta^\top \mathbf{x}_i\mathbf{x}_i^\top\theta - 2y_i\langle\mathbf{x}_i,\theta\rangle\right) + \frac{\lambda}{2}\|\theta\|_2^2 + \langle\boldsymbol{b}_t,\theta\rangle + \theta^\top B_t\theta}_{J_t^{\mathsf{priv}}(\theta)}$, 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 $\texttt{GetSumBias}\left(\mathcal{T}_{\mathsf{bias}}, t\right)$, 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 $\texttt{GetSumCov}\left(\mathcal{T}_{\mathsf{cov}}, t\right)$. 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.
- $\widetilde{\theta}_{t+1} \leftarrow \arg\min_{\theta\in\mathcal{C}} \underbrace{\sum_{i=1}^{t}\left(\theta^\top \mathbf{x}_i\mathbf{x}_i^\top\theta - 2y_i\langle\mathbf{x}_i,\theta\rangle\right) + \frac{\lambda}{2}\|\theta\|_2^2}_{J_t^{\mathsf{np}}(\theta)}$.

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,

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

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

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

(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 (Shalev-Shwartz et al., 2009) 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 (Shalev-Shwartz et al., 2009) and (11), we have that w.p. at least $1-\beta$ over the randomness of the algorithm,

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

(12) immediately implies the following:

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

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 (Shalev-Shwartz et al., 2009))**.** *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 (Mironov, 2017).

Theorem D.1 (Privacy guarantee for DP-FTRL-TR)**.** *If* $\|\nabla_\theta \ell(\theta; d)\|_2 \le L$ *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* $\left(\alpha, \frac{\alpha E L^2 \lceil\lg(n)\rceil}{2\sigma^2}\right)$*-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* $\varepsilon \le 2\ln(1/\delta)$*.*

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* $\|\nabla_\theta \ell(\theta; d)\|_2 \le L$ *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* $\left(\alpha, \frac{\alpha E L^2 \lceil\lg(nE)\rceil}{2\sigma^2}\right)$*-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* $\varepsilon \le 2\ln(1/\delta)$*.*

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 $\left\|\sum_{e\in[E]}\nabla_\theta \ell_e(\theta; d)\right\|_2 \le EL$ 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 (Reddi et al., 2020).

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.

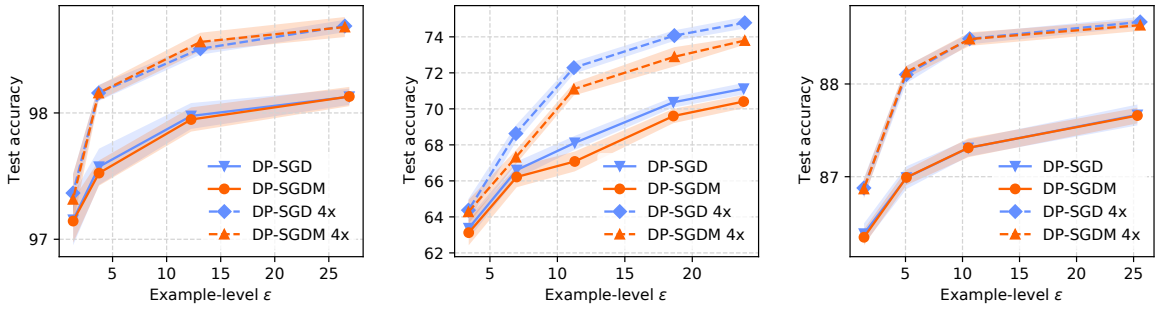
Layer	Parameters
Convolution $\times 2$	32 filters of 3×3, strides 1
Max-Pooling	2×2, stride 2
Convolution $\times 2$	64 filters of 3×3, strides 1
Max-Pooling	2×2, stride 2
Convolution $\times 2$	128 filters of 3×3, strides 1
Max-Pooling	2×2, stride 2
Fully connected	128 units
Softmax	-

(c) Model architecture for StackOverflow. (Reddi et al., 2020)

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

Figures 3 and 4 show a comparison between the original and the momentum versions of DP-SGD (denoted by "DP-SGD" and "DP-SGDM") and DP-FTRL-vanilla (denoted as "DP-FTRL-vanilla" and "DP-FTRLM-vanilla"), respectively, 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-vanilla always outperforms DP-FTRL-vanilla.

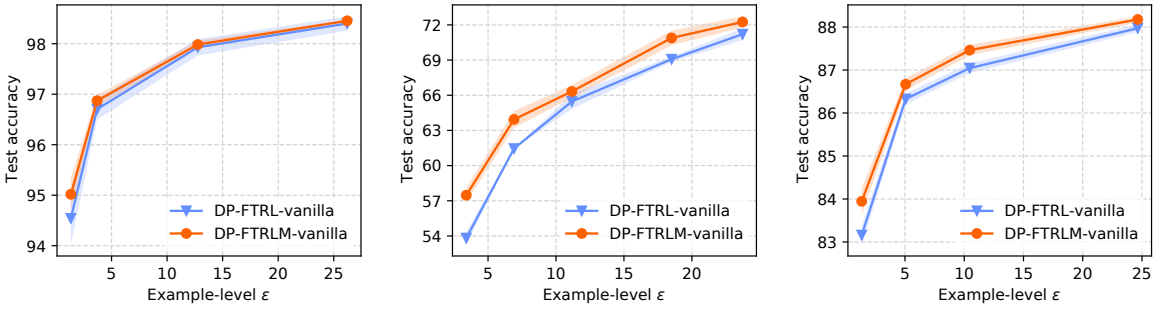


(a) MNIST, DP-SGD(M) (b) CIFAR-10, DP-SGD(M) (c) EMNIST, DP-SGD(M)

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

The experiments in Table 3 and Figure 5 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 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 [9]. We then run the experiment with the specific set of hyperparameters for five times to estimate mean and standard deviation of the accuracy.

[9]The accuracy for StackOverflow next word prediction task excludes the end of sequence symbol and the out of vocabulary symbol following (Reddi et al., 2020). The hyperparameters tuning range are described in Appendix F.1.

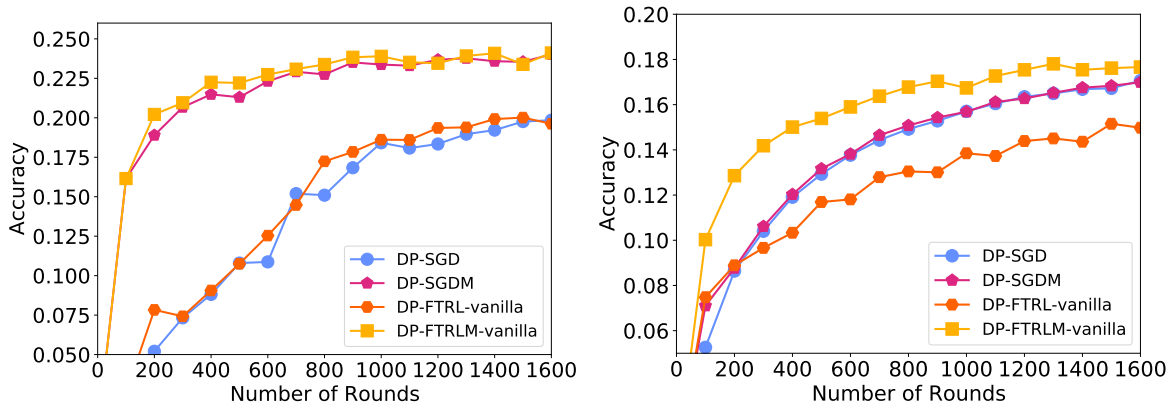


(a) MNIST, DP-FTRL(M)-vanilla (b) CIFAR-10, DP-FTRL(M)-vanilla (c) EMNIST, DP-FTRL(M)-vanilla

Figure 4. Final test accuracy vs. privacy (example-level ε) for various noise multipliers. DP-FTRLM-vanilla outperforms DP-FTRL-vanilla.

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. Vanilla tree aggregation (Dwork et al., 2010) is used in DP-FTRLM. The momentum variant DP-FTRLM performs better than DP-FTRL.



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

Figure 5. Training curves show validation accuracy of StackOverflow. The curve of the best validation accuracy out of the five runs is presented. Vanilla tree aggregation (Dwork et al., 2010) is used in DP-FTRL-vanilla and DP-FTRLM-vanilla. The momentum variant converges faster and performs better.

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.

E.3. Efficient Tree Aggregation

Figure 6 shows a comparison between the efficient ("FTRLM") (Honaker, 2015) and the original version ("FTRLM-vanilla") (Dwork et al., 2010) of FTRLM for the three centralized example-level DP image classification tasks. We can see clearly that the efficient version always outperforms the vanilla version.

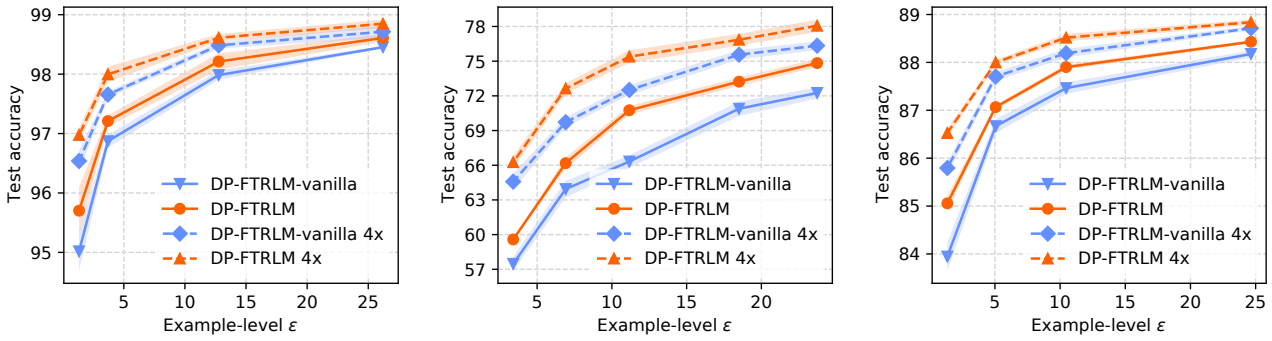


Figure 6. Comparison of two variants of DP-FTRL with efficient tree aggregation ("DP-FTRLM") and vanilla tree aggregation ("DP-FTRLM-vanilla").

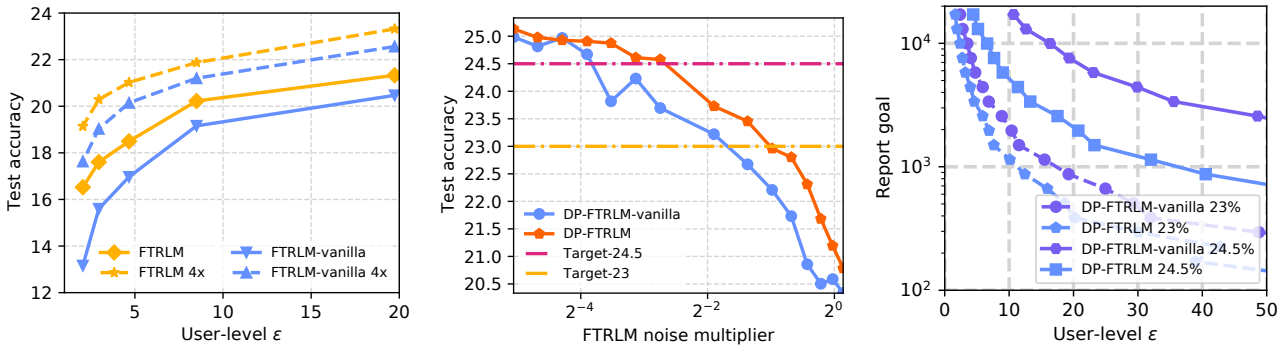
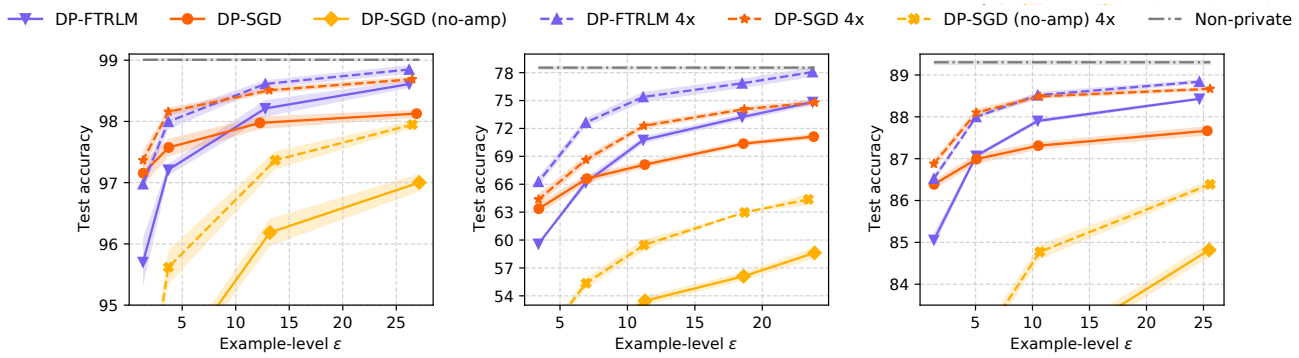


Figure 7. Comparison of two variants of DP-FTRL with efficient tree aggregation (Honaker, 2015) and vanilla tree aggregation (Dwork et al., 2010) on StackOverflow for (a) test accuracy under different privacy epsilon; (b) test accuracy with various noise multipliers; (c) relationship between user-level privacy ε (when $\delta \approx 1/\text{population}$) and computation cost (report goal) for two fixed accuracy targets (see legend).

Figure 7 shows the advantage of the efficient tree aggregation algorithm in the StackOverflow simulation for the federated learning setting. In Figure 7b, to meet the targeted StackOverflow test accuracies (23%, 24.5%), the noise multipliers for DP-FTRLM can increase from (0.268, 0.067) to (0.387, 0.149) after implementing the efficient tree aggregation (Honaker, 2015). The noise multipliers are used to generate Figure 7c.



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

Figure 8. Final test accuracy vs. ε. The lines show the mean and standard deviation over 5 runs. The full plot of Figure 1.

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 (Papernot et al., 2020a).

The parameter search for non-private baseline is the same as that for the DP algorithms. We use regular SGD (with 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 (Reddi et al., 2020). The one layer LSTM described in (Reddi et al., 2020) 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

In Figure 8, 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 highly depends on the batch size and the number of training examples, which affects 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 11, which shows for a specific accuracy α, the two ε-batch curves 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 (resp. 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 (resp. 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

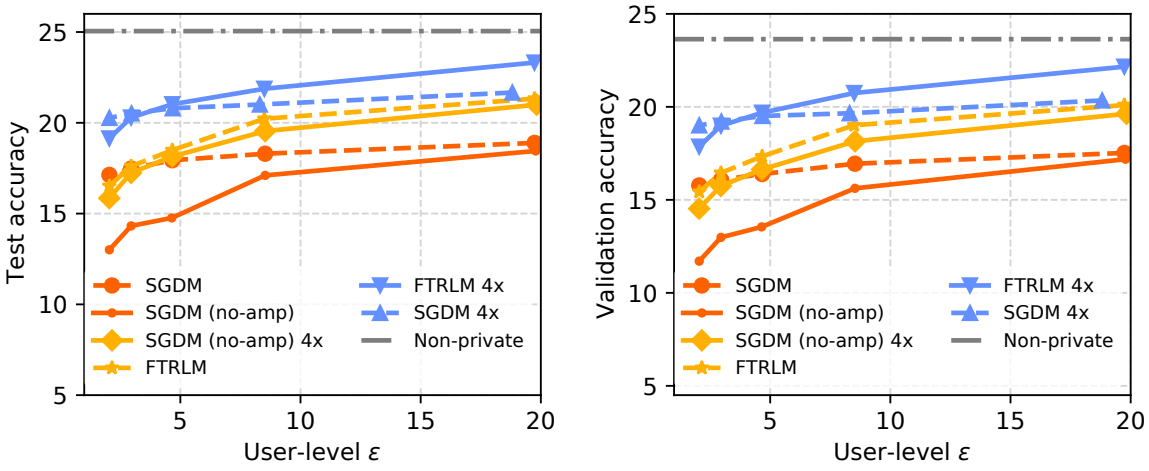


Figure 9. Test and Validation accuracy for the StackOverflow next word prediction task under different privacy epsilon by varying noise multipliers.

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 9). DP-FTRLM performs better than DP-SGDM when the epsilon is relatively large, but performs worse when the epsilon is small ($\varepsilon < 2.95$ in Table 4). 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 (Balle et al., 2020). 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 (McMahan et al., 2017b), and more discussion on this front is in Appendix G.

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	20.10	21.33	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	19.01	20.22	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	17.34	18.49	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	16.45	17.60	6.21	0.1	0.5	1
DP-SGDM	2.05	15.78	17.13	0.7	0.3	0.5	0.3
DP-FTRLM	2.04	15.43	16.52	8.83	0.3	0.5	0.3

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

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 10 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.

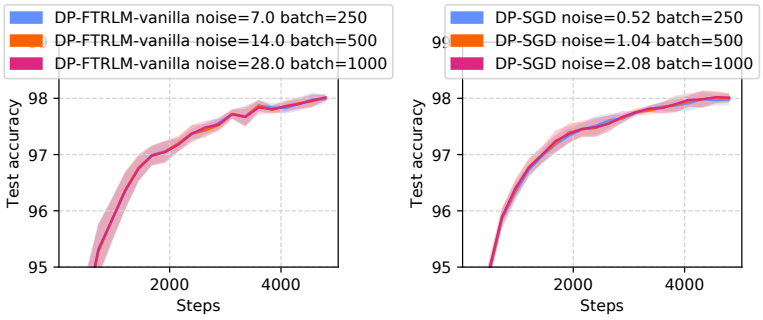
As we have confirmed that scaling the batch size and noise together does not affect the accuracy, in Figure 11, we plot the ε value versus batch size b such that σ/b is a fixed value. We can see that as batch size grows, FTRL achieves better privacy than DP-SGD at the same level of accuracy.

G.2. Details of Hyperparameter Tuning

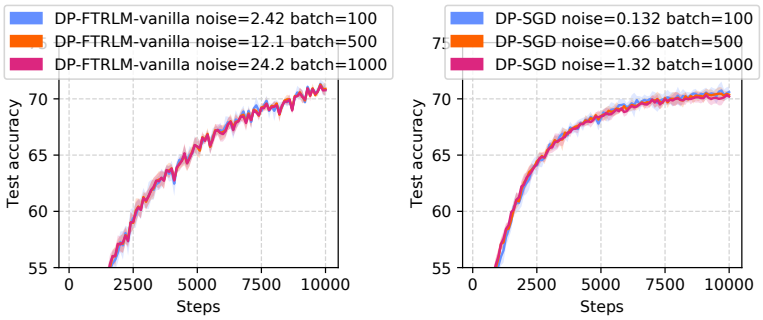
In Appendix F.3, a significant amount of noise has to be added in both DP-FTRLM and DP-SGDM to achieve nontrivial privacy epsilons, which leads to undesired accuracy degradation. For example, the test accuracy of DP-FTRLM on StackOverflow dataset decreases from 25.15% when $\varepsilon = \infty$ to 20.22% 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 (Bonawitz et al., 2019)) 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 12. We then tune the server learning rate from $\{0.3, 1, 3\}$ for each noise multiplier.

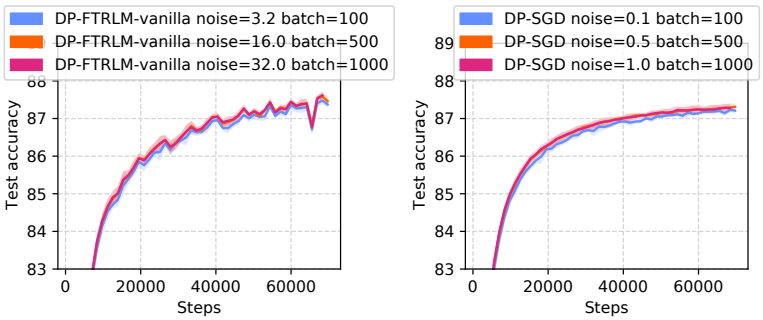
Grid search for the largest noise multiplier to meet the target. We use a grid of ten noise multipliers between 0 ($\varepsilon = \infty$, test accuracy=24.89) and 0.3 ($\varepsilon = 18.89$, test accuracy=18.89) for DP-SGDM, and between 0 ($\varepsilon = \infty$, test accuracy=25.15) and 1.13 ($\varepsilon = 19.74$, test accuracy=21.33) 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 0.149 (with server learning rate 3) for DP-FTRLM.



(a) MNIST.

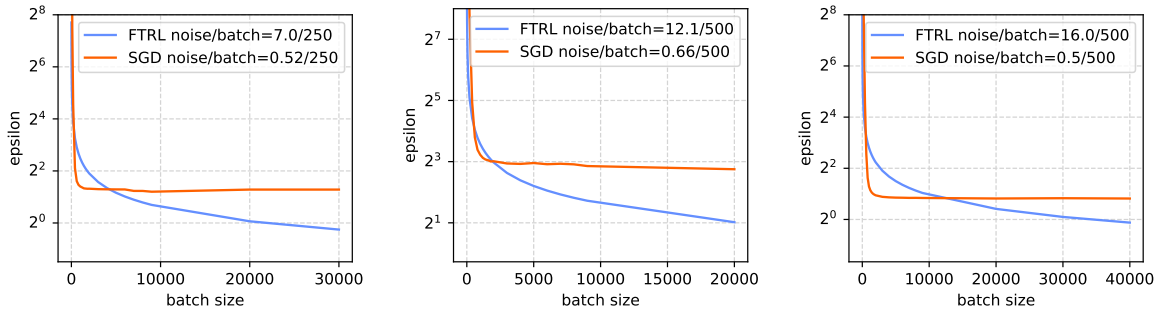


(b) CIFAR-10.



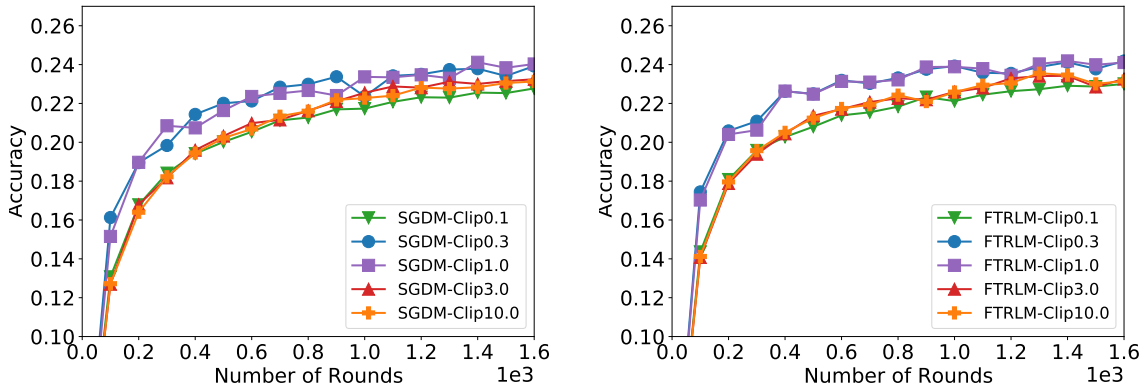
(c) EMNIST (ByMerge).

Figure 10. 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 11. ε vs. batch size. According to Figure 10, 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 12. Training curve of the best validation accuracy under various clip norm for StackOverflow.

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 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 (McMahan et al., 2017b) while the privacy guarantee is improved. The validation accuracy of simulation performance with two different report goals for StackOverflow is presented in Figure 13. The noise multiplier 0.149 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 24.51% for DP-FTRLM. We then run the same experiments with report goal of 1000, and proportionally increase the corresponding noise multiplier to be 1.49 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 24.67% 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 $3.42e5$ [10]. 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 14 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[11], 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 $\varepsilon > 3.2$ at $\delta = 10^{-6}$ for utility target 24.5%, and nearly all $\varepsilon \in (0, 50]$ for utility target 23%.

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

[11] 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.

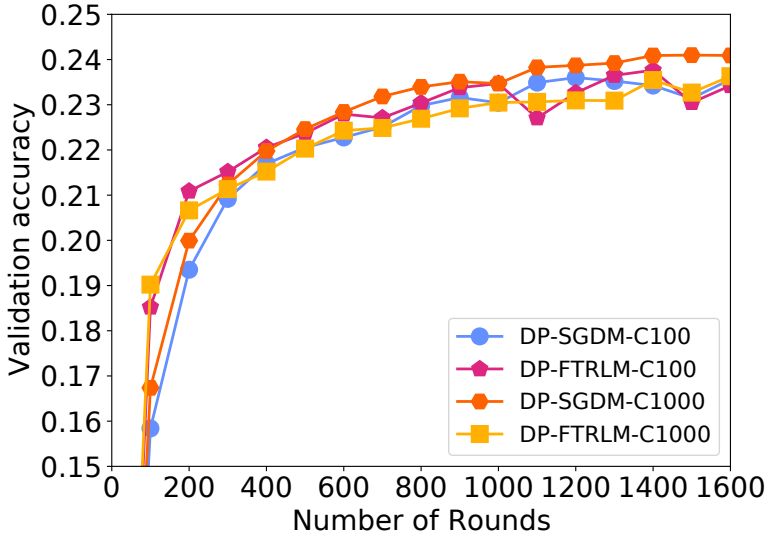


Figure 13. Training curves of validation accuracy for DP-SGDM and DP-FTRLM for StackOverflow for report goal 100 and 1000 (suffix C100 and C1000 in the legend). DP-FTRLM with restart (see Appendix D) is used when report goal is 1000 (less than five epochs of data). Simultaneously increasing noise multiplier 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, 0.149, 0.07, and 1.49, respectively. The corresponding test accuracy are 24.73%, 24.51%, 25.19% and 24.67%. 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	364.74	1e-6	0.149	100	3.42e5
DP-SGDM	7.71e4	1e-6	0.07	1000	3.42e5
DP-FTRLM	33.80	1e-6	1.49	1000	3.42e5
DP-SGDM	10.16	1e-6	23.97	3.42e5	3.42e5
DP-FTRLM	10.53	1e-6	7.53	5.06e3	3.42e5
DP-FTRLM	4.57	1e-6	24.29	1.63e4	3.42e5
DP-SGDM	8.98	1e-6	0.67	9.56e3	1e6
DP-FTRLM	8.24	1e-6	5.73	3.85e3	1e6
DP-SGDM	4.17	1e-6	1.20	1.71e4	1e6
DP-FTRLM	4.00	1e-6	15.29	1.03e4	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.

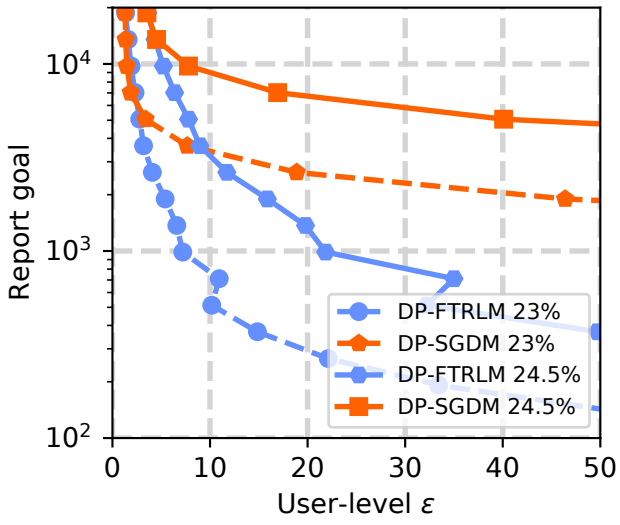


Figure 14. Relationship between privacy ε (when $\delta = 1/\text{population}$) and report goal for a fixed accuracy target for DP-FTRLM and DP-SGDM on the StackOverflow dataset with a hypothetically larger population of 1M users.

Table 6. Parameters for the image classification experiment in Figure 1 (and the full version Figure 8) and Figure 10. Clipping norm is 1.0. The "learning rate" reported for FTRL(M) is λ. For FTRL(M), we used "-v" as an abbreviation for "-vanilla".

(a) MNIST

$b = 250$ 20 epochs	FTRL-v/FTRLM-v/FTRL	noise	4.0	7.0	20.0	50.0
		ε	26.21	12.76	3.70	1.34
		learning rate	1.0/10.0/10.0	2.0/20.0/10.0	5.0/50.0/50.0	10.0/100.0/100.0
	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-v/FTRLM	noise	8.0	14.0	40.0	100.0
		ε	26.21	12.76	3.70	1.34
		learning rate	5.0/5.0	10.0/10.0	20.0/20.0	50.0/50.0
	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-v/FTRLM-v/FTRL	noise	10	12.1	18.1	27	50
		ε	23.73	18.51	11.17	6.91	3.40
		learning rate	2.0/20.0/10.0	2.0/20.0/20.0	5.0/50.0/20.0	5.0/50.0/50.0	20.0/100.0/50.0
	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-v/FTRLM	noise	20	24.2	36.2	54	100
		ε	23.73	18.51	11.17	6.91	3.40
		learning rate	10.0/5.0	10.0/10.0	20.0/10.0	20.0/20.0	50.0/50.0
	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-v/FTRLM-v/FTRL	noise	8.0	16.0	30.0	100.0
		ε	24.64	10.46	5.06	1.35
		learning rate	2.0/20.0/10.0	2.0/20.0/20.0	5.0/50.0/50.0	10.0/100.0/100.0
	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-v/FTRLM	noise	16.0	32.0	60.0	200.0
		ε	24.64	10.46	5.06	1.35
		learning rate	10.0/5.0	20.0/10.0	20.0/20.0	50.0/50.0
	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 10. Clipping norm is 1.0. The "learning rate" reported for FTRL(M) is λ.

Data	Batch size		FTRLM-vanilla	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