Recycling Scraps: Improving Private Learning by Leveraging Checkpoints

Virat Shejwalkar*
Google Deepmind
vshejwalkar@google.com

Arun Ganesh*
Google Research
arunganesh@google.com

Rajiv Mathews
Google Deepmind

Yarong Mu
Google

Shuang Song
Google Deepmind

Om Thakkar
OpenAI†

Abhradeep Thakurta
Google Deepmind

Xinyi Zheng
Google

Abstract

DP training pipelines for modern neural networks are iterative and generate multiple checkpoints. However, all except the final checkpoint are discarded after training. In this work, we propose novel methods to utilize intermediate checkpoints to improve prediction accuracy and estimate uncertainty in DP predictions. First, we design a general framework that uses aggregates of intermediate checkpoints *during training* to increase the accuracy of DP ML techniques. Specifically, we demonstrate that training over aggregates can provide significant gains in prediction accuracy over the existing state-of-the-art for StackOverflow, CIFAR10 and CIFAR100 datasets. For instance, we improve the state-of-the-art DP StackOverflow accuracies to 22.74% (+2.06% relative) for $\varepsilon = 8.2$, and 23.90% (+2.09%) for $\varepsilon = 18.9$. Furthermore, these gains magnify in settings with periodically varying training data distributions. We also demonstrate that our methods achieve relative improvements of 0.54% and 62.6% in terms of utility and variance, on a proprietary, production-grade pCVR task. Lastly, we initiate an exploration into estimating the uncertainty (variance) that DP noise adds in the predictions of DP ML models. We prove that, under standard assumptions on the loss function, the sample variance from last few checkpoints provides a good approximation of the variance of the final model of a DP run. Empirically, we show that the last few checkpoints can provide a reasonable lower bound for the variance of a converged DP model. Crucially, all the methods proposed in this paper operate on *a single training run* of the DP ML technique, thus incurring no additional privacy cost.

Keywords

Differential privacy, Prediction uncertainty, Deep learning

*The first two authors contributed equally.
†Work done when the author was at Google.

Proceedings on Privacy Enhancing Technologies 2025(2), 607–628
© 2025 Copyright held by the owner/author(s).
https://doi.org/10.56553/popets-2025-0079

1 Introduction

Machine learning models can unintentionally memorize sensitive information about the data they were trained on, which has led to numerous attacks that extract private information about the training data [7, 20–22, 40, 41, 74]. To address such privacy risks, literature has introduced various approaches to privacy-preserving ML [61, 73, 81]. In particular, iterative techniques like differentially private stochastic gradient descent (DP-SGD) [3, 12, 56, 77] and DP Follow The Regularized Leader (DP-FTRL) [48] have become the state-of-the-art for training DP neural networks.

The accuracy-variance trade-off is a central problem in machine learning. By accuracy, we mean the primary evaluation metric of a model on train/test data, e.g., accuracy for CIFAR10 and StackOverflow, and AUC-loss (i.e., 1 - AUC) for pCVR. Techniques like DP-SGD and DP-FTRL involve the operation of *per-example gradient clipping* and calibrated Gaussian noise addition in each training step, which makes this trade-off even trickier to understand in DP ML [78]. In this work, we propose using checkpoints (i.e., intermediate model iterates) to improve on both fronts of the problem.

Our contributions at a glance: First, we design a general framework that (adaptively) uses aggregates of intermediate checkpoints (i.e., the intermediate iterates of model training; see Definition 2.3) to increase the accuracy of DP ML techniques. Next, we provide a method to estimate the uncertainty (variance) that DP noise adds to DP ML training. Crucially, we attain both these goals with *a single training run* of the DP technique, thus incurring no additional privacy cost (ignoring the cost of hyperparameter tuning; see Section 3.3).

We emphasize that improved accuracy and uncertainty estimation are complementary goals - a practical model is more useful if it achieves higher accuracy/lower train loss, but also if we can use uncertainty estimates to avoid making risky predictions caused purely by noise. However, while both goals can be achieved in the same training run, for ease of exposition we will describe our techniques for them separately. In the following, we provide the details of our contributions, and place them in the context of prior works.

Increasing accuracy using checkpoints aggregates (Sections 3 and 4): While the privacy analyses of state-of-the-art DP ML techniques allow releasing/using all the training checkpoints, prior works [3, 5, 6, 10, 33, 34, 37, 48, 54, 56, 65, 82, 88, 94] use only the final model output by the DP algorithm for establishing benchmarks or for deployment in practice [53, 67]. To our knowledge, [26] is the only prior work that re-uses intermediate checkpoints to increase the accuracy of DP-SGD and notes non-trivial accuracy gains.

In this work, we propose to use checkpoints aggregates at training and/or inference time to improve DP-ML performances. Specifically, we propose *the first* general adaptive training framework, ATF (Algorithm 2), to adaptively train over intermediate checkpoints aggregates to improve DP-ML performances; we give two concrete instantiations, $\mathsf{EMA}_{\mathsf{tr}}$ and $\mathsf{UTA}_{\mathsf{tr}}$, of the framework. For the inference time, we propose three methods, $\mathsf{UTA}_{\mathsf{inf}}$, OPA and OMV, to aggregate checkpoints' parameters or outputs to improve DP-ML performance. Performances of our methods depend on certain key hyperparameters, e.g., on EMA coefficient β for EMA based methods, hence we propose hyperparameters tuning algorithms for proposed aggregations, demonstrate how they improve DP-ML performances, and also extensively discuss the privacy implications of such hyperparameters tuning. Empirically, we demonstrate significant performance gains on for three standard benchmarks (StackOverflow, CIFAR10, CIFAR100) and a proprietary, production-grade pCVR dataset. It is worth noting that DP state-of-the-art has repeatedly improved over the years since the foundational techniques from [3] for CIFAR and [56] for StackOverflow, hence any consistent improvements are instrumental in advancing the state of DP ML.

In all our experiments, we note that $\mathsf{UTA}_{\mathsf{tr}}$ achieves the best performances. Specifically, it achieves the state-of-the-art prediction accuracy of 22.74% at $\varepsilon = 8.2$ for StackOverflow (i.e., 2.09% relative gain over DP-FTRL from [48])[1], and 57.51% at $\varepsilon = 1$ for CIFAR10 (i.e., 2.7% relative gain over DP-SGD as per [26]), respectively. For CIFAR100, we first improve the DP-SGD baseline of [26] by warm-starting DP training on CIFAR100 from the EMA checkpoint of the ImageNet pre-training pipeline instead of its last checkpoint as in [26]. We improve DP-SGD performance by 5% and 3.2% (absolute) for ε 1 and 8, respectively. $\mathsf{UTA}_{\mathsf{tr}}$ further improves the accuracy on CIFAR100 by 0.67% to 76.18% at $\varepsilon = 1$, i.e., 0.89% relative gain over our improved CIFAR100 DP-SGD baseline. Note that, all of our proposed training/inference checkpoints aggregations improve over the corresponding DP-SGD baselines, however for conciseness and clarity, we do not provide all the results.

Next, we show that these benefits further magnify in more practical settings with periodically varying training data distributions. For instance, we note relative accuracy gains of 2.64% and 2.82% for ε of 18.9 and 8.2, respectively, for StackOverflow over DP-FTRL baseline in such a setting. Finally, we experiment with a proprietary, production-grade pCVR dataset [25, 27] and show that at $\varepsilon = 6$, $\mathsf{UTA}_{\mathsf{tr}}$ improves AUC-loss (i.e., 1 - AUC) by 0.54% (relative) over the DP-SGD baseline. Note that such an improvement is considered very significant in the context of ads ranking.

Theoretically, we show in Theorem 3.2 that for standard training regimes, the excess empirical risk of the final checkpoint of DP-SGD is $\log(n)$ times more than that of the weighted average of the past k checkpoints, where n is the size of dataset. It is interesting to theoretically analyze the use of checkpoint aggregations during training, which we leave as future work.

[1]These improvements are notable since there are $10k$ classes in StackOverflow data.

Uncertainty quantification using intermediate checkpoints (Section 5): There are various sources of randomness in an ML training pipeline [4], e.g., choice of initial parameters, dataset, batching, etc. This randomness induces uncertainty in the predictions made using such ML models. In critical domains, e.g., medical diagnosis, self-driving cars and financial market analysis, failing to capture the uncertainty in such predictions can have undesirable repercussions. We refer the reader to [43] for a survey of uncertainty estimation and its applications. DP learning adds an additional source of randomness by injecting noise at every training round. Hence, it is paramount to quantify reliability of the DP models, e.g., by quantifying the uncertainty in their predictions [18, 66].

As prior work, [49] develops finite sample confidence intervals but for the simpler Gaussian mean estimation problem. Various methods exist for uncertainty quantification in ML-based systems [14, 39, 45, 52, 59, 60, 69, 79, 86]. However, these methods either use specialized (or simpler) model architectures to facilitate uncertainty quantification, or are not directly applicable to quantify the uncertainty in DP ML due to DP noise. For example, a common way of uncertainty quantification [11, 16, 35, 62] that we call the *independent runs* method, needs k independent (bootstrap) runs of the ML algorithm. However, repeating a DP ML algorithm multiple times can incur significant privacy and computation costs.

To this end, *for the first time we quantify the uncertainty that DP noise adds* to DP training procedure *using only a single training run*. We propose to use the last k checkpoints of a single run of a DP ML algorithm as a proxy for the k final checkpoints from independent runs. This does not incur *any additional privacy cost* to the DP ML algorithm. Furthermore, it is useful in practice as it does not incur additional training compute, and can work with any algorithm having intermediate checkpoints. Finally, it doesn't require changing the underlying model or algorithm, unlike some other methods for uncertainty estimation (e.g., the use of Bayesian neural networks [90]).

Theoretically, we consider using (a rescaling of) the sample variance of a statistic $f(\theta)$ at checkpoints $\theta_{t_1}, \ldots, \theta_{t_k}$ as an estimator of the variance of any convex combination of $f(\theta_{t_i})$, i.e., any weighted average of the statistics at the checkpoints, and give a bound on the bias of this estimator. As expected, our bound on the error decreases as the "*burn-in*" time t_1 and the time between checkpoints t_2 both increase. An upshot of this analysis is that getting k nearly i.i.d. checkpoints requires fewer iterations than running k independent runs of t_1 iterations. *In turn, under a fixed privacy constraint, using the sample variance of the checkpoints can provide more samples and thus tighter confidence intervals than the independent runs method*; see the remark in Section 5 for details.

Empirically, we show that our method provides reasonable lower bounds on the uncertainty quantified using the more accurate (but privacy and computation intensive) method that uses independent runs. For instance, we show that for DP-FTRL trained StackOverflow, the 95% confidence widths for the scores of the predicted labels computed using independent runs method (no budget split)[2] are always within a factor of 2 of the widths provided by our method for various privacy levels and number of bootstrap samples.

[2]Thus, a superior baseline by not splitting the privacy budget among the independent runs.

While we compute the variance in regards to a fixed prediction function, we believe our estimator can be used to obtain DP parameter confidence intervals for traditional statistical estimators (*e.g.*, linear regression). We leave this direction for future exploration.

2 Background and Preliminaries

In this section, we briefly introduce the background on machine learning, privacy leakages in machine learning models, differential privacy and deep learning with differential privacy.

2.1 Machine Learning

In this paper, we consider machine learning (ML) models used for image classification and language next-word-prediction tasks. We use *supervised machine learning* for both the types of tasks and briefly review it below.

Let $f_\theta : \mathbb{R}^d \mapsto \mathbb{R}^k$ be a ML classifier (e.g., neural network) with d input features and k classes, which is parameterized by θ. For a given example $\mathbf{z} = (\mathbf{x}, y)$, $f_\theta(\mathbf{x})$ is the classifier's confidence vector for k classes and the predicted label is the corresponding class which has the largest confidence score, i.e., $\hat{y} = \arg\max_i f_\theta(\mathbf{x})$. The goal of supervised machine learning is to learn the relationship between features and labels in given *labeled* training data D^l_{tr} and generalize this ability to unseen data. The model learns this relationship using empirical risk minimization (ERM) on the training set D^l_{tr}, where the risk is measured in terms of a certain loss function, e.g., cross-entropy loss:

$$\min_\theta \frac{1}{|D^l_{tr}|} \sum_{\mathbf{z} \in D^l_{tr}} l(f_\theta, \mathbf{z}))$$

Here $|D^l_{tr}|$ is the size of the labeled training set and $l(f_\theta, \mathbf{z})$ is the loss function. When clear from the context, we use f instead of f_θ, to denote the target model.

2.2 Privacy Leakage in ML Models

ML models generally require large amounts of training data to achieve good performances. This data can be of sensitive nature, e.g., medical records and personal photographs, and without proper precautions, ML models may leak sensitive information about their private training data. Multiple previous works have demonstrated this via various *inference* attacks, e.g., membership inference, property or attribute inference, model stealing, and model inversion. Below, we review these attacks.

Consider a target model f_θ trained on D_{tr} and a target sample $(\mathbf{x}, y)$. Membership inference attacks [8, 71, 75] aim to infer whether the target sample $(\mathbf{x}, y)$ was used to train the target model, i.e., whether $(\mathbf{x}, y) \in D_{tr}$. Property or attribute inference attacks [57, 76] aim to infer certain attributes of $(\mathbf{x}, y)$ based on model's inference time representation of $(\mathbf{x}, y)$. For instance, even if f_θ is just a gender classifier, $f_\theta(\mathbf{x})$ may reveal the race of the person in $\mathbf{x}$. Model stealing attacks [63, 83] aim to reconstruct the parameters θ of the original model f_θ based on black-box access to f_θ, i.e., using $f_\theta(\mathbf{x})$. Model inversion attacks [40] aim to reconstruct the whole training data D_{tr} based on white-box, i.e., using θ, or black-box, i.e., using $f_\theta(\mathbf{x})$, access to model.

2.3 Deep Learning with Differential Privacy

Differential privacy [29–31] is a notion to quantify the privacy leakage from the outputs of a data analysis procedure and is the gold standard for data privacy. It is formally defined as below:

Definition 2.1 (Differential Privacy). *A randomized algorithm $\mathcal{M}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ preserves (ε, δ)-differential privacy iff for any two neighboring datasets $D, D' \in \mathcal{D}$ and for any subset $S \subseteq \mathcal{R}$ we have:*

$$\mathbf{Pr}[\mathcal{M}(D) \in S] \le e^\varepsilon \, \mathbf{Pr}[\mathcal{M}(D') \in S] + \delta \tag{1}$$

where ε is the privacy budget *and δ is the* failure probability.

Rényi Differential Privacy (RDP) is a commonly-used relaxed definition for differential privacy.

Definition 2.2 (Rényi Differential Privacy (RDP) [58]). *A randomized algorithm $\mathcal{M}$ with domain $\mathcal{D}$ is (α, ε)-RDP with order $\alpha \in (1, \infty)$ if and only if for any two neighboring datasets $D, D' \in \mathcal{D}$:*

$$\begin{aligned} &D_\alpha(\mathcal{M}(D) || \mathcal{M}(D')) \\ &:= \frac{1}{\alpha - 1} \log \mathop{\mathbb{E}}_{\delta \sim \mathcal{M}(D')} [(\frac{Pr[\mathcal{M}(D) = \delta]}{Pr[\mathcal{M}(D') = \delta]})^\alpha] \le \varepsilon \end{aligned} \tag{2}$$

There are two key properties of DP algorithms that will be useful in our *composition* and *post-processing*. Below we briefly review these two properties specifically for the widely-used Rényi-DP definition, but they apply to all the DP algorithms.

Lemma 1 (Adaptive Composition of RDP [58]). *Consider two randomized mechanisms $\mathcal{M}_1$ and $\mathcal{M}_2$ that provide (α, ε_1)-RDP and (α, ε_2)-RDP, respectively. Composing $\mathcal{M}_1$ and $\mathcal{M}_2$ results in a mechanism with $(\alpha, \varepsilon_1 + \varepsilon_2)$-RDP.*

Lemma 2 (Post-processing of RDP [58]). *Given a randomized mechanism that is (α, ε)-RDP, applying a randomized mapping function on it does not increase its privacy budget, i.e., it will result in another (α, ε)-RDP mechanism.*

2.3.1 *Differentially Private ML Algorithms We Use.*

Several works have used differential privacy in traditional machine learning to protect the privacy of the training data [13, 23, 38, 51, 92]. We use two of the commonly-used algorithms for DP deep learning: DP-SGD [2], and DP-FTRL [48]. At a high level, to update the model in each training round, DP-SGD first samples a minibatch of examples uniformly at random, clips the gradient of each example to limit the sensitivity of a gradient update, and then adds independent Gaussian noise to gradients that is calibrated to achieve the desired DP guarantee. In contrast, in each training round, DP-FTRL takes a minibatch of examples (no requirement of sampling), clips each example's gradient to limit sensitivity, and adds correlated Gaussian noise calibrated to achieve the desired DP guarantee.

Algorithm 1 details the full-batch DP-GD algorithm. We focus on using checkpoints, defined as follows:

Definition 2.3. *A checkpoint is any intermediate model iterate θ_i of a training algorithm, e.g. as defined in DP-GD (Algorithm 1).*

We emphasize that *all* model iterates are defined as checkpoints, but we will not use all checkpoints in our methods. For training, we

Algorithm 1 DP Gradient Descent (DP-GD)

```
$\theta_0 \leftarrow \mathbf{0}^p$.
for $t \in [T]$ do
    $\theta_{t+1} \leftarrow \Pi_C\left(\theta_t - \eta_t\left(\nabla\mathcal{L}(\theta_t; D) + b_t\right)\right)$, where $b_t \sim \mathcal{N}\left(0, \frac{L^2 T}{2n\rho}\mathbb{I}_{p\times p}\right)$, and $\Pi_C(\cdot)$ being the $\ell_2$-projection onto the set $C$.
end for
```

will usually use a suffix of the checkpoints, i.e. $\theta_{T-k}, \theta_{T-k+1}, \ldots, \theta_T$. For uncertainty estimation, we will use a subset of k checkpoints with unspecified indices $t_1 < t_2 < \ldots < t_k$; here we will denote checkpoints as θ_{t_i} rather than θ_i.

3 Using Checkpoint Aggregates to Improve Accuracy of Differentially Private ML

In this section, we first detail our novel and general *adaptive aggregation training framework* that leverages past checkpoints (recall a checkpoint is just an intermediate model iterate θ_t) during training, and provide two instantiations of it. We also design four checkpoint aggregation methods that can be used for inference over a given sequence of checkpoints. Finally, we provide a theoretical analysis for improved privacy-utility trade-offs due to some of the checkpoint aggregations.

Why can we post-process intermediate DP ML checkpoints?: Before delving into the details of our checkpoints aggregation methods, it is useful to note that the privacy analyses for the DP algorithms we consider in this paper, i.e., DP-SGD [2] and DP-FTRL [48], use the adaptive composition (Lemma 1) across training rounds. This implies that all the intermediate checkpoints are also DP, which allows us to release of all intermediate checkpoints computed during training. Furthermore, as all checkpoints are DP, due to the post-processing property of DP (Lemma 2), one can process/use these checkpoints without incurring additional privacy cost.

3.1 Using Checkpoint Aggregations for Training

Algorithm 2 describes our general adaptive aggregation training framework. Apart from the parameters needed to run the DP algorithm $\mathcal{A}$, it uses a *checkpoint aggregation* function f_{AGG} to compute an aggregate checkpoint $\theta^{\mathsf{AGG}}_{t+1}$ from the checkpoints $(\theta_{t+1}, \theta_t, \ldots, \theta_0)$ at each step t. Consequently, $\mathcal{A}$ uses $\theta^{\mathsf{AGG}}_{t+1}$ for its next training step. Note that Algorithm 2 has two hyperparameters: (1) τ that decides when to start training over the past checkpoints aggregate, and (2) parameter p specific to f_{AGG} which we detail below, along with f_{AGG}s. Due to the post-processing property of DP, using f_{AGG} does not incur any additional privacy cost. Though our framework can incorporate any custom f_{AGG}, we present two natural instantiations for f_{AGG} and extensively evaluate them.

***Exponential Moving Average (EMA)*:** Our first proposal uses an EMA function to aggregate all the past checkpoints at training step t. Starting from the latest checkpoint, EMA assigns exponentially decaying weights to each of the previous checkpoints. At step t, EMA maintains a moving average θ^{EMA}_t that is a weighted average

Algorithm 2 Our adaptive aggregation training framework.

```
Input: Iterative DP ML algorithm $\mathcal{A}$, private dataset $D$, initial model $\theta_0$, number of training steps $T$, checkpoints aggregation function $f_{\mathsf{AGG}}$ and its parameter $p$ (EMA coefficient $\beta$ for $\mathsf{EMA}_{\mathsf{tr}}$ and number of last $k$ checkpoints for $\mathsf{UTA}_{\mathsf{tr}}$), the step to start training over past aggregate $\tau$
$\theta^{\mathsf{AGG}}_0 = \theta_0$.
for $t = 0$ to $T$ do
    if $t \geq \tau$ then
        $\theta_{t+1} \leftarrow \mathcal{A}(\theta^{\mathsf{AGG}}_t; D)$.
        $\theta^{\mathsf{AGG}}_{t+1} = f_{\mathsf{AGG}}(\{\theta_{t+1}, \theta_t, \ldots, \theta_0\}, p)$.
    else
        $\theta_{t+1} \leftarrow \mathcal{A}(\theta_t; D)$.
    end if
end for
Return $\theta^{\mathsf{AGG}}_{t+1}$
```

of $\theta^{\mathsf{EMA}}_{t-1}$ and the latest checkpoint, θ_t . This is formalized as follows:

$$\theta^{\mathsf{EMA}}_t = (1-\beta_t)\cdot\theta^{\mathsf{EMA}}_{t-1} + \beta_t\cdot\theta_t \tag{3}$$

***Uniform Tail Averaging (UTA)*:** Our second proposal uses a UTA function to aggregate past k checkpoints. Specifically, for step t, UTA computes the *parameter-wise* mean of the past $\min\{t+1, k\}$ checkpoints. We formalize this as:

$$\theta^{\mathsf{UTA}}_t = \frac{1}{\min\{t+1, k\}}\sum_{i=\max\{0, t-(k-1)\}}^{t}\theta_i \tag{4}$$

3.2 Using Checkpoint Aggregations for Inference

In many scenarios, e.g., where a DP ML technique has been applied to release a sequence of checkpoints, checkpoint aggregation functions can be used as post-processing functions over the released checkpoints to reduce bias of the technique at inference time. In this section, we design various aggregation methods towards this goal.

We note that [17, 80] have used EMA (Equation 3) to improve the performance of ML techniques at inference time in non-private settings. De et al. [26] extend EMA to DP-SGD, but use EMA coefficients β suggested from non-private settings; we denote this EMA baseline by $\mathsf{EMA}_{\mathsf{baseline}}$. However, as we will show in Section 4, even a coarse-grained tuning of β provides significant accuracy gains in DP settings. To highlight the crucial difference with the instantiation in Section 3.1, we use $\mathsf{EMA}_{\mathsf{tr}}$ to denote when we use aggregation adaptively in training (Algorithm 2), and $\mathsf{EMA}_{\mathsf{inf}}$ to denote when we use the aggregation only for inference. Since UTA (Equation 4) can be applied as an aggregation at inference time, we similarly define $\mathsf{UTA}_{\mathsf{tr}}$ and $\mathsf{UTA}_{\mathsf{inf}}$.

Outputs aggregation functions: So far, our aggregation functions have focused on aggregating parameters of intermediate checkpoints. Next, we design two aggregation functions that, given a sequence of checkpoints $\theta_i, i \in [t]$, compute a function of the outputs of the checkpoints and use it for making predictions.

***Output Predictions Averaging (OPA)*:** For a given test sample $\mathbf{x}$, OPA first computes prediction vectors $f_{\theta_i}(\mathbf{x})$ of the last k checkpoints, i.e., checkpoints from steps $\in [t-(k-1), t]$, averages the prediction vectors, and computes argmax of the average vector as the final output label. We formalize OPA as follows:

$$\hat{y}_{\text{opa}}(\mathbf{x}) = \text{argmax}\Big(\frac{1}{k}\sum_{i=t-(k-1)}^{t} f_{\theta_i}(\mathbf{x})\Big) \tag{5}$$

***Output Labels Majority Vote (OMV)*:** For a given test sample $\mathbf{x}$, OMV computes output prediction labels, i.e., argmax $f_{\theta_i}(\mathbf{x})$ for the last k checkpoints. Finally, it outputs the majority label among the k labels (breaking ties arbitrarily) for inference. We formalize OMV as follows:

$$\hat{y}_{\text{omv}}(\mathbf{x}) = \text{Majority}\big(\text{argmax}(f_{\theta_i}(\mathbf{x}))_{i=t-(k-1)}^{t}\big) \tag{6}$$

3.2.1 Improved Excess Risk via Tail Averaging. Results from [72] can be used to demonstrate how a family of checkpoint aggregations, which includes $\mathsf{UTA}_{\mathsf{inf}}$ (Section 3.2), provably improves the privacy/utility trade-offs compared to that of the last checkpoint of DP-(S)GD. To formalize the problem, we define the following notation: Consider a data set $D = \{d_1, \dots, d_n\}$ and a loss function $\mathcal{L}(\theta; D) = \frac{1}{n}\sum_{i=1}^{n} \ell(\theta; d_i)$, where each of the loss function ℓ is convex and L-Lipschitz in the first parameter, and $\theta \in C$ with $C \subseteq \mathbb{R}^p$ being a convex constraint set. We analyze the following variant of DP-GD (Algorithm 1), which is guaranteed to be ρ-zCDP defined below. Note that using [19], it is easy to convert the privacy guarantee to an (ε, δ)-DP guarantee. Moreover, while our analytical result is for DP-GD (due to brevity), it extends to DP-SGD with mild modifications to the proof.

Definition 3.1 (zCDP [19]). *A randomized algorithm $M : \mathcal{D}^* \to \mathcal{Y}$ is ρ-zero-concentrated differentially private (zCDP) if, for all neighbouring datasets $D, D' \in \mathcal{D}^*$ (i.e., datasets differing in one data sample) and all $\alpha \in (1, \infty)$, we have*

$$\mathrm{D}_\alpha\left(M(D) \| M(D')\right) \le \rho\alpha$$

where $\mathrm{D}_\alpha\left(M(D) \| M(D')\right)$ is the α-Rényi divergence between the distribution of $M(D)$ and $M(D')$.

We will provide the utility guarantee for this algorithm by directly appealing to the result of [72]. For a given $\alpha \in (0, 1)$, $\mathsf{UTA}_{\mathsf{inf}}$ corresponds to the average of the last αT models, i.e.,

$$\theta_t^{\mathsf{UTA}} = \frac{1}{\alpha T}\sum_{t=(1-\alpha)T+1}^{T} \theta_t \tag{7}$$

One can also consider *polynomial-decay averaging* (PDA) with parameter $\gamma \ge 0$, defined as follows:

$$\theta_t^{\mathsf{PDA}} = \left(1 - \frac{\gamma+1}{t+\gamma}\right)\theta_{t-1}^{\mathsf{PDA}} + \frac{\gamma+1}{t+\gamma}\cdot\theta_t \tag{8}$$

For $\gamma = 0$, PDA matches $\mathsf{UTA}_{\mathsf{inf}}$ over all iterates. As γ increases, PDA places more weight on later iterates; in particular, if $\gamma = cT$, the averaging is similar to $\mathsf{EMA}_{\mathsf{inf}}$ (Section 3.2), since as $t \to T$ the decay parameter $\frac{\gamma+1}{t+\gamma}$ approaches a constant $\frac{c}{c+1}$. In that sense, PDA can be viewed as a method interpolating between $\mathsf{UTA}_{\mathsf{inf}}$ and $\mathsf{EMA}_{\mathsf{inf}}$. From [72], we can derive the following bounds on the different methods:

Theorem 3.2. *There exists a choice of learning rate η_t and the number of time steps T in DP-GD (Algorithm 1) such that the following hold for $\alpha = \Theta(1)$:*

$$\mathbb{E}\left[\mathcal{L}\left(\theta_{\text{priv}}^{\mathsf{UTA}}; D\right)\right] - \min_{\theta \in C}\mathcal{L}(\theta; D) = O\left(\frac{L\,\|C\|_2\sqrt{p}}{n\rho}\right)$$

and

$$\mathbb{E}\left[\mathcal{L}(\theta_T; D)\right] - \min_{\theta \in C}\mathcal{L}(\theta; D) = O\left(\frac{L\,\|C\|_2\sqrt{p}\log(n)}{n\rho}\right).$$

Furthermore, for $\gamma = \Theta(1)$, we have,

$$\mathbb{E}\left[\mathcal{L}\left(\theta_T^{\mathsf{PDA}}; D\right)\right] - \min_{\theta \in C}\mathcal{L}(\theta; D) = O\left(\frac{L\,\|C\|_2\sqrt{p}}{n\rho}\right).$$

Proof. These bounds build on Theorems 2 and 4 of [72]. If we choose $T = \lceil n\rho \rceil$ and set η_t appropriately, the proof of Theorem 2 [72] implies the following for θ_T^{UTA}:

$$\mathbb{E}\left[\mathcal{L}\left(\theta_T^{\mathsf{UTA}}; D\right)\right] - \min_{\theta \in C}\mathcal{L}(\theta; D) = O\left(\frac{L\,\|C\|_2\sqrt{p}}{n\rho}\log\left(\frac{1}{\alpha}\right)\right).$$

Setting $\alpha = \Theta(1)$ gives the theorem's first part, and $\alpha T = 1$, i.e., $1/\alpha = T = \lceil n\rho \rceil$ gives the second. The third follows from modifying Theorem 4 of [72] for the convex case (see the end of Section 4 of [72] for details). ◻

Theorem 3.2 implies that the excess empirical risk for θ_T is higher by factor of $\log(n)$ in comparison to θ_T^{UTA} and θ_T^{PDA}. For step size selections typically used in practice (e.g., fixed or inverse polynomial step sizes), the last iterate will suffer from the extra $\log(n)$ factor, and we do not know how to avoid it. Furthermore, Harvey et al. [44] showed that this is unavoidable in the non-private, high probability regime. Jain et al. [46] show that for carefully chosen step sizes, the logarithmic factor can be removed, and Feldman et al. [36] extend this analysis to a DP-SGD variant with varying batch sizes. Unlike those methods, averaging can be done as post-processing of DP-SGD outputs, rather than a modification of the algorithm.

3.3 Hyperparameters Tuning for Our Aggregations

Performances of our aggregations depend on certain hyperparameters. Hence, in this section, we first discuss advantages and disadvantages of these hyperparameters' values, followed by the specific methodology we use to tune the hyperparameters, and finally discuss privacy implications of such tuning.

3.3.1 Significance of values of key hyperparameters. There are two key hyperparameters in our aggregations. In $\mathsf{EMA}_{\mathsf{inf}}$ and $\mathsf{EMA}_{\mathsf{tr}}$, EMA coefficient β sets the weights of the checkpoints. Specifically larger β gives higher weight to newer checkpoints which are generally better than previous checkpoints hence we tune β from as low as 0.5.

The number k of past checkpoints aggregated affects the performances of the rest of the training and inference aggregations. Very large k includes contribution of checkpoints from early training while very small k may ignore good checkpoints, both of which may hurt the performance of the final aggregate. Therefore, we tune k in a fairly wide range starting from $k = 3$ up to $k = 200$.

Algorithm 3 Hyperparameters tuning for training aggregations.

```
Input: Adaptive training algorithm $\mathcal{A}^{\text{Ada}}$ (Algorithm 2) with aggregation function $f_{\text{AGG}}$ and its hyperparameter $p$, range of hyperparameters $\{p, \tau\}$ for grid search $R_{p,\tau}$, validation set $D_v$, $T$ training steps, Initial $\theta_0$.
Initialize: $\text{Acc}_{max} \leftarrow 0$, $\theta_{best} \leftarrow \theta_0$, $\{p_{best}, \tau_{best}\} \leftarrow \{1, 0\}$.
for $\{p, \tau\}$ in $R_{p,\tau}$ do
    Run $\mathcal{A}^{\text{Ada}}$ for $T$ steps with $f_{\text{AGG}}, p, \tau$ as detailed in Algorithm 2
        $\theta_T^{\text{Ada}} \leftarrow \mathcal{A}^{\text{Ada}}(f_{\text{AGG}}, p, \tau, \theta_0)$
    Compute accuracy of the output checkpoint on validation set:
        $\text{Acc}_{\text{Ada}} = \text{Acc}(\theta_T^{\text{Ada}}, D_v)$
    if $\text{Acc}_{\text{Ada}} > \text{Acc}_{max}$ then
        $\text{Acc}_{max} \leftarrow \text{Acc}_{\text{AGG}}$, $\theta_{best} \leftarrow \theta_T^{\text{Ada}}$, $\{p_{best}, \tau_{best} \leftarrow \{p, \tau\}$
    end if
end for
Return $\theta_{best}, p_{best}, \tau_{best}$
```

3.3.2 Training aggregations. We use a simple grid-search strategy to tune hyperparameters as detailed in Algorithm 3. Note that there are two hyperparameters to tune: aggregation parameters p and step to start training over past aggregate τ. For EMA_{tr}, p in Algorithm 3 is the EMA coefficient β in (3), and we tune $\beta \in \{0.5, 0.6, 0.7, 0.8, 0.85, 0.9, 0.95, 0.99, 0.999, 0.9999\}$ for all datasets. For StackOverflow we fix $\tau = 0$ while for CIFAR10 we tune $\tau \in \{100, 200, 500, 1000, \dots \tau^*\}$ where τ^* is largest multiple of 500 smaller than total number of steps T; we tune $\tau \in \{50, 100, \dots, 250\}$ for CIFAR100. For UTA_{tr}, p in Algorithm 3 is the number of k past checkpoints to aggregate. For CIFAR10/CIFAR100 we tune $k \in \{3, 5, 10, ..., 100\}$, for pCVR we tune $k \in \{3, 5\}$ and for StackOverflow we tune $k \in \{3, 5, 10, 20, ..., 200\}$ for UPA_{tr}. Finally note that, in case of StackOverflow, we use inference aggregation after producing all intermediate checkpoints using training aggregations. So we follow the hyperparameter tuning strategies for training and inference aggregations in sequence.

3.3.3 Inference aggregations. Our simple grid-search strategy to tune hyperparameters is detailed in Algorithm 4. For EMA_{inf}, p in Algorithm 4 is the EMA coefficient β in (3). De et al. [26] simply use β that works the best in non-private settings. However, tuning $\beta \in \{0.85, 0.9, 0.95, 0.99, 0.999, 0.9999\}$, we observe that the best β for private and non-private settings need not be the same (Table 1). For instance, for CIFAR10, for ε of 1 and 8, EMA_{inf} coefficient of 0.95 and 0.99 perform the best and outperform 0.9999 by 0.6% and 0.3%, respectively. Hence, we advise future works to perform tuning of EMA coefficient. Full results are given in Table 1. For UTA_{inf}, OPA and OMV, p in Algorithm 4 is the number of last checkpoints k to aggregate. We tune k in the same range as in training aggregations. Table 7 in Appendix B shows results for CIFAR10, where we observe non-trivial gains due to tuning k.

Algorithm 4 Hyperparameters tuning for inference aggregations.

```
Input: Intermediate checkpoints $(\theta_{T-1}, \dots \theta_0)$ from $T$ training steps, checkpoints aggregation function $f_{\text{AGG}}$ and its hyperparameter $p$, range of $p$ for grid search $R_p$, validation set $D_v$.
Initialize: $\text{Acc}_{max} \leftarrow 0$, $\theta_{best} \leftarrow \theta_{T-1}$, $p_{best} \leftarrow 1$.
for $p$ in $R_p$ do
    Compute aggregated checkpoint
        $\theta_T^{\text{AGG}} = f_{\text{AGG}}(\{\theta_{T-1}, \dots \theta_0\}, p)$
    Compute accuracy of aggregated checkpoint on validation set:
        $\text{Acc}_{\text{AGG}} = \text{Acc}(\theta_T^{\text{AGG}}, D_v)$
    if $\text{Acc}_{\text{AGG}} > \text{Acc}_{max}$ then
        $\text{Acc}_{max} \leftarrow \text{Acc}_{\text{AGG}}$, $\theta_{best} \leftarrow \theta_T^{\text{AGG}}$, $p_{best} \leftarrow p$
    end if
end for
Return $\theta_{best}, p_{best}$
```

Table 1: Tuning the EMA coefficient can provide significant gains in accuracy over the default value of 0.9999 from [26] implying the need to tune EMA coefficients for each different privacy budget to achieve the best performances. Results below are for original CIFAR10 dataset.

Privacy level	EMA coefficient			
	0.9	0.95	0.99	0.999 ([26])
$\varepsilon = 8$	79.41	79.35	**79.41**	79.16
$\varepsilon = 1$	56.59	**56.61**	56.06	56.05

3.3.4 Privacy cost of hyperparameter tuning. Consistent with past works (e.g. [26, 48]) we will ignore the privacy cost of hyperparameter tuning in our experiments. If instead one wants to account for hyperparameter tuning cost, one can use private hyperparameter selection algorithms. We show that using these algorithms, the privacy cost of the added hyperparameters of our methods over hyperparameter tuning of baseline methods is minimal. To show this, we consider the hyperparameter selection algorithm of [64]: We sample a number of trials, for each trial sample a random hyperparameter setting and run the training algorithm, and then pick the best training run based on some metric (e.g. performance on a public test set). As our baseline, we consider the hyperparameter sweep used by [26], which sweeps over learning rate, clip norm, and augmentation multiplicity for a total of 72 grid points. For e.g. our inference aggregation methods, we would additionally sweep over 6 values of β, resulting in a grid size of $6 \cdot 72 = 432$ instead. In Figure 1 we plot the increase in privacy of a Gaussian mechanism due to this hyperparameter tuning algorithm for 72 or 432 expected trials using the algorithm of [64]. We see that at a baseline of $\varepsilon = 1$ for a single training run, hyperparameter tuning over the grid of [26] increases ε to 2.62. Additionally sweeping over β causes the increases in ε due to hyperparameter tuning to change to 2.98, an increase of 13.8%. When $\varepsilon = 8$ for a single training run, the grid of [26] increases ε to 16.08, and adding β to the grid further increases it to 18.34, a 14.1% increase. In other words, additionally tuning β does not cause the privacy to significantly degrade compared to vanilla DP-SGD with hyperparameter tuning.

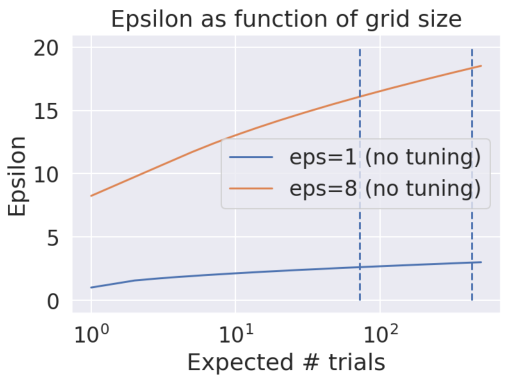

Figure 1: We plot the value of ε accounting for hyperparameter tuning of a $(1, 10^{-5})$-DP and $(8, 10^{-5})$-DP Gaussian mechanism, using the algorithm of [64] with the number of trials sampled from a Poisson distribution. We include two vertical lines at 72 and 432, corresponding to a grid for "vanilla" DP-SGD, and the product of this grid with a sweep over β.

4 Empirical Evaluation

In this section, we first describe experimental setup, followed by experiments in a user-level and sample-level DP settings.

4.1 Experimental Setup

4.1.1 Datasets and ML Settings. We evaluate our checkpoints aggregation algorithms on three benchmark datasets (StackOverflow, CIFAR10, CIFAR100) and one proprietary production-grade dataset (pCVR) in two different settings.

StackOverflow: StackOverflow [47] is a natural-language dataset containing questions and answers from StackOverflow forum. We use it to train a model for next word prediction task. StackOverflow is a *user-keyed dataset*, i.e., all the samples in the data are owned by some users. It is a large dataset containing training data of total of 342,477 users and over 135M samples. The original test data contains data of 204,088 users; following [68], we sample 10,000 users for validation data. Following [68], we use vocabulary of top-10,000 words from StackOverflow data.

We use simulated federated learning (FL) [55] to train on StackOverflow data. In each FL round, a central server (model trainer) broadcasts a global model to all users, users share gradient updates that they compute using the model and their local dataset. The central server then aggregates all user updates and updates the global model to be used for the following FL rounds.

CIFAR Datasets: We experiment with CIFAR10 and CIFAR100 datasets. CIFAR10 (CIFAR100) [50] is a 10-class (100-class) image classification task and contains 60,000 32 × 32 color (RGB) images (50,000 images as training set and 10,000 images as test set). We use centralized ML for CIFAR10 (CIFAR100) training, i.e., when model trainer collects all data in one place and trains a model on it.

pCVR (Predicted Conversion Rate) Dataset: This is a proprietary, production-grade dataset (also used in [25, 27]), where each example corresponds to an ad click, and the task is to predict whether a conversion takes place after the click, which is commonly referred as predicted conversion rate (pCVR). As users' clicking and conversion information is highly sensitive, such data needs to be protected with differential privacy. We use centralized ML for training, similar to CIFAR datasets. This dataset contains significantly more examples, by orders of magnitude, than the aforementioned datasets.

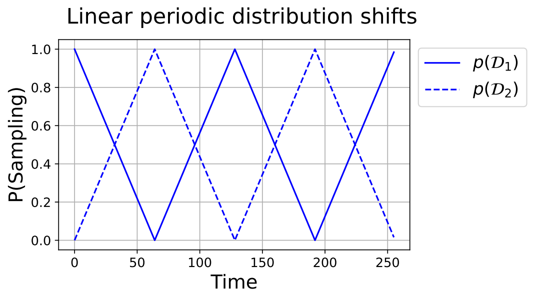

Figure 2: Probability of sampling users or samples from two periodically shifting distributions $\mathcal{D}_{\{1,2\}}$.

4.1.2 Periodic Distribution Shift (PDS) Settings. The distribution of data sampled from the datasets discussed above is almost uniform throughout the training; we call such datasets ***original datasets***. However, in many real-world settings, e.g., in FL, the training data distribution may vary over time. Zhu et al. [93] demonstrate the adverse impacts of distribution shifts in training data on the performances of resulting FL models. Due to their practical significance, we consider settings where the training data distribution models *diurnal* variations, i.e., it is a function of two oscillating distributions (see Figure 2 for an example). Such a scenario commonly occurs in FL training, e.g., when a model is trained with client devices participating from two significantly different time zones.

Following [93], we consider a setting where training data is a combination of clients/samples drawn from two disjoint data distributions which oscillate over time (Figure 2). Here, the probabilities of sampling at time t are: $p(\mathcal{D}_1, t) = \left|2\frac{t \bmod T}{T} - 1\right| \; p(\mathcal{D}_2, t) = (1 - p(\mathcal{D}_1, t))$, where T is the period of oscillation of $\mathcal{D}_{\{1,2\}}$.

Simulating periodic distribution shifting settings: To simulate such periodically shifting distribution for StackOverflow, we use $\mathcal{D}_1$ with only questions and $\mathcal{D}_2$ with only answers from users. Then, we draw clients from $\mathcal{D}_{\{1,2\}}$. Apart from data distribution, the rest of experimental setup is the same as before. We use test and validation data same as for the original StackOverflow setting. To simulate PDS CIFAR10/CIFAR100, we use $\mathcal{D}_{\{1,2\}}$ such that $\mathcal{D}_1$ and $\mathcal{D}_2$ respectively contain the data from even and odd classes of the original data; the rest of the sampling strategy is the same as described in Section 4.1.2.

4.1.3 Model Architectures and Training Details. Below we detail the model architectures, DP ML algorithms, and various hyperparameters we use to obtain our results.

Note that, for each of the tasks we evaluate, we select the state-of-the-art DP ML algorithm as the baseline algorithm and demonstrate improvements on top of the performances of such state-of-the-art DP ML algorithms. For instance, we use DP-FTRL for StackOverflow task as it provides state-of-the-art performance on StackOverflow;

DP-SGD does not perform well on StackOverflow hence we omit it from StackOverflow experiments. For the same reason, we use DP-SGD for the rest of the tasks.

StackOverflow training: For StackOverflow, we follow the state-of-the-art DP training in [28, 48] and train a one-layer LSTM using DP-FTRL with momentum in Tensorflow Federated framework [1] for $\varepsilon \in \{8.2, 18.9\}$, which corresponds to ρ-zCDP with $\rho \in \{1.08, 4.31\}$, respectively. We process 100 users in each FL round and train for total of 2,000 rounds. For experiments with DP, we fix the privacy parameter δ to 10^{-6} for StackOverflow ensuring that $\delta < n^{-1}$, where n is the number of users in StackOverflow. Since StackOverflow data is naturally keyed by users, the privacy guarantees here are at user-level, in contrast to the example-level privacy for CIFAR10.

Tables 8 and 9 provide the hyperparameters we use for training aggregations ($\mathsf{UTA}_{\mathsf{tr}}$, $\mathsf{EMA}_{\mathsf{tr}}$) using DP-FTRL.

CIFAR10 training: Following the setup of the state-of-the-art DP-SGD training in [26], we train a WideResNet-16-4 with depth 16 and width 4 using DP-SGD [3] in JAXline [9] for $\varepsilon \in \{1, 8\}$. We fix clip norm to 1, batch size to 4096 and augmentation multiplicity to 16 as in [26]. For experiments with DP, we fix the privacy parameter δ to 10^{-5} on CIFAR10 ensuring that $\delta < n^{-1}$, where n is the number of examples in CIFAR10. Here the DP guarantee is at sample-level.

For training on CIFAR10, we use the state-of-the-art DP-SGD parameters from [26] as follows: we set learning rate and noise multiplier, respectively, to 2 and 10 for $\varepsilon = 1$ and to 4 and 3 for $\varepsilon = 8$. We stop the training when the intended privacy budget exhausts. All the hyperparameters we use to generate the results of Table 3 are in Table 10.

CIFAR100 training: Similarly to [26], for CIFAR100, we use Jaxline [15] and use DP-SGD to *fine-tune the last, classifier layer of* a WideResNet with depth 28 and width 10 that is pre-trained on entire ImageNet data. We fix clip norm to 1, batch size to 16,384 and augmentation multiplicity to 16. Then, we set learning rate and noise multiplier, respectively, to 3.5 and 21.1 for $\varepsilon = 1$ and to 4 and 9.4 for $\varepsilon = 8$. For periodic distribution shifting (PDS) CIFAR100, we set learning rate and noise multiplier, respectively, to 4 and 21.1 for $\varepsilon = 1$ and to 5 and 9.4 for $\varepsilon = 8$. We stop the training when privacy budget exhausts. Setup for training aggregations is the same as for CIFAR10 above; hyperparameters used to generate results in Table 4 are in Table 11.

pCVR Training: We employ a multi-encoder model architecture, where each encoder is responsible for encoding a specific class of features (e.g., ads features). We consider sample level privacy with $\varepsilon = 6$ and $\delta = \frac{1}{n}$, where n is the number of examples, as these are the parameters that are of production requirement.

The model is trained with logistic loss and is measured by the test *AUC loss* (i.e., 1 - AUC), as is commonly done for pCVR tasks [25, 27]. In real-world advertising scenarios, the pCVR models' outputs (i.e., the predicted conversion probability) are often passed directly to downstream models for calculating final ad bids, instead of being converted to binary predictions. Therefore, we use AUC-loss instead of other commonly used classification metrics, such as accuracy. For the same reason, Majority voting ($\mathsf{OMV}_{\mathsf{tr}}$) is not applicable for this task.

We adopt a two-stage hyperparameter-tuning strategy for DP-SGD. We first tune the batch size, number of steps, clip norm, and learning rate for baseline DP-SGD, and then, with the above fixed, tune the hyperparameters in Section 3.3. This is done primarily due to the significant training cost associated with pCVR.

4.2 Experiments with User-level Privacy on StackOverflow Dataset

In this section, we evaluate efficacy of our aggregation methods in a *user-level DP* setting. Specifically, we first perform experiments with original StackOverflow data described in Section 4.1.1, then describe a more real-world setting with periodically shifting distribution (PDS) of dataset and present results for the PDS setting.

4.2.1 Aggregation Methods We Use With Original StackOverflow. We evaluate two *training* and four *inference* aggregation methods. For training aggregations, we consider $\mathsf{EMA}_{\mathsf{tr}}$ and $\mathsf{UTA}_{\mathsf{tr}}$ methods (Section 3.1). For inference aggregations, we consider $\mathsf{EMA}_{\mathsf{inf}}$, $\mathsf{UTA}_{\mathsf{inf}}$, OPA, and OMV methods (Section 3.2). For $\mathsf{UTA}_{\mathsf{tr}}$, we first use our adaptive training framework (ATF) with f_{UTA} as f_{AGG}, as described in Section 3.1. Then we use our post-processing based inference framework on top of the checkpoints generated by ATF to produce the results in Tables 2 and 3. We similarly produce results for $\mathsf{EMA}_{\mathsf{tr}}$ in Tables 2 and 3. Following [26, 80], we use a warm-up schedule for the EMA coefficient as:

$$\beta_t = \min(\beta, (1+t)/(10+t))$$

We note that for EMA, one can further optimize this schedule, but this will require a larger grid so we opt not to. Tuning only β or k makes our tuning compute friendly (and would keep the additional privacy cost due to private hyperparameter selection reasonably small if we were to use it; see Section 3.3). All our results are average of 5 runs of each setting.

4.2.2 Results for Original StackOverflow. In the rest of the paper, the tables present results for the final training round T, while plots show results over the last k rounds for some $k \ll T$. Due to large size of StackOverflow test data, we provide plots for accuracy on validation data and tables with accuracy on test data.

Table 2 presents the accuracy gains in StackOverflow for $\varepsilon \in \{\infty, 18.9, 8.2\}$ due to our training and inference aggregations. We observe that **our training aggregation $\mathsf{UTA}_{\mathsf{tr}}$ always provides the maximum accuracy gains**. Specifically, for ε of ∞, 18.9, and 8.2, $\mathsf{UTA}_{\mathsf{tr}}$ provides relative (absolute) accuracy improvement over the baseline (DP-FTRL with momentum) of 2.97% (0.75%), 2.09% (0.49%), and 2.06% (0.46%) respectively. The corresponding relative (absolute) accuracy improvement over $\mathsf{EMA}_{\mathsf{baseline}}$ (i.e., EMA over baseline with EMA coefficients as per [26]) are 1.05% (0.27%), 1.48% (0.45%), and 1.43% (0.32%) respectively. Note that while De et al. [26] do not have StackOverflow experiments, we provide results for $\mathsf{EMA}_{\mathsf{baseline}}$ using EMA and EMA coefficient β suggested in [26].

Finally, in the leftmost two plots in Figure 3, we focus on the inference aggregations since they just post-process the checkpoints of the state-of-the-art baseline run. First, note that **all of inference aggregations significantly outperform the baseline ($\mathsf{UTA}_{\mathsf{inf}}$ performs the best among all inference aggregations)**. Second, due to DP noise, the accuracy of baseline DP checkpoints

Table 2: Test accuracy gains due to checkpoints aggregations for original and PDS StackOverflow. We present techniques from prior works (DP-FTRL baseline [28, 48], and $\text{EMA}_{\text{baseline}}$ [26]) in blue.

DP	Baseline	Training Aggregations		Inference Aggregations				
(ε)	(No Agg)	EMA_{tr}	UTA_{tr}	$\text{EMA}_{\text{baseline}}$	EMA_{inf}	UTA_{inf}	OPA	OMV
StackOverflow; DP-FTRL; user-level privacy								
∞	25.24 ± 0.16	25.72 ± 0.02	**25.98 ± 0.01**	25.71 ± 0.02	25.79 ± 0.01	25.81 ± 0.02	25.79 ± 0.01	25.78 ± 0.01
18.9	23.41 ± 0.08	23.56 ± 0.02	**23.90 ± 0.02**	23.55 ± 0.03	23.63 ± 0.01	23.84 ± 0.01	23.60 ± 0.02	23.57 ± 0.02
8.2	22.28 ± 0.08	22.43 ± 0.04	**22.74 ± 0.04**	22.42 ± 0.04	22.54 ± 0.02	22.70 ± 0.03	22.57 ± 0.04	22.52 ± 0.04
Periodic Distribution Shifting (PDS) StackOverflow; DP-FTRL; user-level privacy								
∞	23.89 ± 0.14	23.97 ± 0.04	**24.26 ± 0.02**	23.86 ± 0.09	23.92 ± 0.12	23.98 ± 0.02	23.87 ± 0.01	23.91 ± 0.07
18.9	21.60 ± 0.13	21.90 ± 0.04	**22.17 ± 0.03**	21.80 ± 0.04	21.82 ± 0.07	22.04 ± 0.11	21.99 ± 0.13	21.95 ± 0.16
8.2	20.24 ± 0.29	20.37 ± 0.06	**20.81 ± 0.05**	20.36 ± 0.07	20.36 ± 0.06	20.75 ± 0.05	20.67 ± 0.03	20.72 ± 0.16

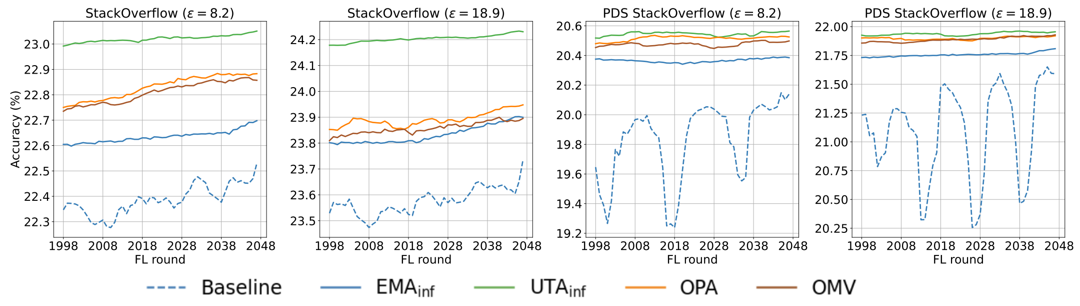

Figure 3: Accuracy gains due to inference aggregations (Section 3.2) for DP-FTRL on original and PDS StackOverflow.

has very high variance across training rounds, which is undesirable in practice. However, we note that all considered inference aggregations significantly reduce such variance while consistently providing gains in accuracy. In other words, **our checkpoints aggregations produce good DP models with high confidence, which is highly desired in practice**. The left plot in Figure 4 presents results for the non-private setting with $\varepsilon = \infty$ and we note similar improvements due to our inference aggregations.

It is worth mentioning that the DP state-of-the-art for the datasets we consider have repeatedly improved over the years since the foundational techniques from [3] for CIFAR-10 and [56] for StackOverflow, so we consider the consistent improvements that our proposed technique provide as significant improvements.

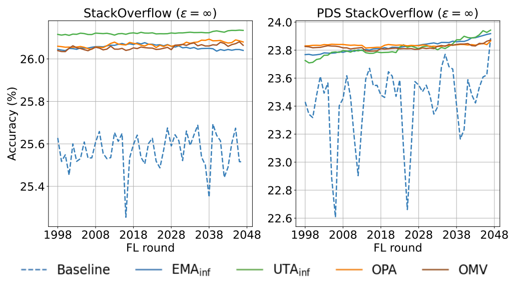

Figure 4: Performances of inference aggregations (Section 3.2) in non-private settings ($\varepsilon = \infty$). We note significant accuracy gains for DP-FTRL on original and PDS StackOverflow even in the non-private settings.

4.2.3 *Results for StackOverflow With Periodic Distribution Shifts.* Last four rows of Table 2 and the rightmost two plots of Figure 3 present accuracy gains for PDS StackOverflow (discussed in Section 4.1.2). **For PDS StackOverflow as well,** UTA_{tr} **always provides the maximum accuracy gains**; specifically for ε of ∞, 18.9, and 8.2, the relative (absolute) accuracy gains due to UTA_{tr} over the DP-FTRL baseline are 1.55% (0.37%), 2.64% (0.57%), and 2.82% (0.57%) respectively. While the relative (absolute) gains over $\text{EMA}_{\text{baseline}}$ are 1.67% (0.42%), 1.7% (0.27%), and 2.21% (0.44%) respectively. The rightmost two plots of Figure 3 show results of using our inference aggregations (Section 3.2) in PDS setting. We note that the variance of accuracy of the baseline DP-FTRL checkpoints is very high for the PDS setting, which is undesirable in practice. However, **our inference aggregations almost completely eliminate the variance in PDS setting, while producing more accurate predictions**.

4.3 Experiments With Sample-level Privacy on CIFAR10 Dataset

In this section, we evaluate efficacy of our aggregation methods (Section 4.2.1) in a *sample-level DP* setting with the original CIFAR10 and CIFAR10 with periodic distribution shifts (PDS).

4.3.1 *Results for Original CIFAR10.* Table 3 and the left-most two plots in Figure 5 present the accuracy gains in CIFAR10 for $\varepsilon \in \{1, 8\}$. **For CIFAR10 as well** UTA_{tr} **provides highest accuracy gains**. Specifically, for ε of 1 and 8, the relative (absolute) accuracy gains due to UTA_{tr} are 8.86% (4.68%) and 3.6% (2.78%) over the DP-SGD baseline, and they are 2.70% (1.51%) and 1.01% (0.8%) over $\text{EMA}_{\text{baseline}}$. Among the inference aggregations, for $\varepsilon = 1$, OPA

Table 3: Test accuracy gains for original and periodic distribution shifting (PDS) CIFAR10. We present techniques from prior works (DP-SGD and $\text{EMA}_{\text{baseline}}$ [26]) in blue.

DP	Baseline	Training Aggregations		Inference Aggregations				
(ε)	(No Agg)	EMA_{tr}	UTA_{tr}	$\text{EMA}_{\text{baseline}}$	EMA_{inf}	UTA_{inf}	OPA	OMV
CIFAR10; DP-SGD; sample-level privacy								
8	77.18 ± 1.46	78.98 ± 0.26	**79.96** ± **0.24**	79.16 ± 0.50	79.41 ± 0.51	79.39 ± 0.52	79.40 ± 0.59	79.34 ± 0.54
1	52.83 ± 2.17	56.24 ± 0.42	**57.51** ± **0.31**	56.00 ± 0.71	56.61 ± 0.91	56.62 ± 0.89	56.68 ± 0.89	56.40 ± 0.69
Periodic Distribution Shifting (PDS) CIFAR10; DP-SGD; sample-level privacy								
8	60.74 ± 1.75	78.18 ± 0.39	**79.19** ± **0.44**	77.99 ± 0.94	78.24 ± 0.92	77.92 ± 0.89	78.27 ± 0.84	77.99 ± 0.94
1	47.13 ± 1.76	54.11 ± 0.63	**55.01** ± **0.48**	54.04 ± 0.81	54.04 ± 0.81	54.35 ± 0.90	54.58 ± 0.82	54.03 ± 1.08

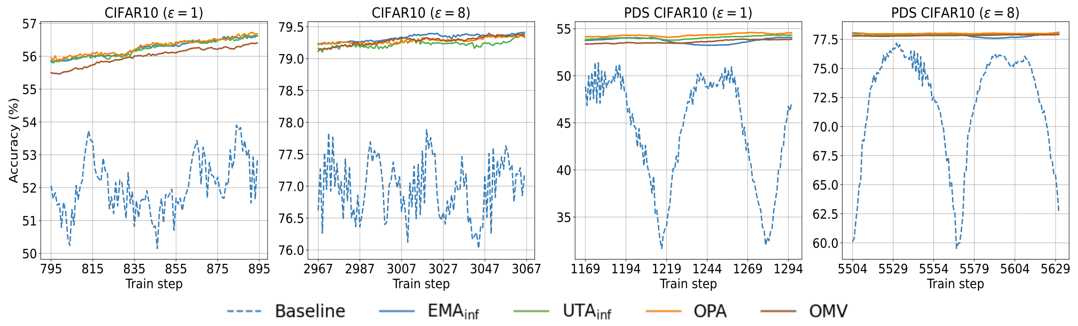

Figure 5: Accuracy gains due to inference aggregation methods (Section 3.2) for DP-SGD on original and PDS CIFAR10.

provides the maximum relative (absolute) accuracy gain of 7.3% (3.85%), while for $\varepsilon = 8$, EMA_{inf} provides maximum gain of 2.9% (2.23%) over the DP-SGD baseline. We note from Figure 5 that **all checkpoints aggregations improve accuracy for all the training steps of DP-SGD for both ε's**. Also note from Figure 5 that, the accuracy of baseline DP-SGD has a high variance across training steps and our inference aggregations significantly reduce this variance.

4.3.2 Results for CIFAR10 With Periodic Distribution Shifts. Section 4.1.2 discusses how we emulate periodic distribution shifting (PDS) CIFAR10 data. Note that to train using DP-SGD on PDS CIFAR10, we set learning rate and noise multiplier, respectively, to 2 and 12 for $\varepsilon = 1$ and to 4 and 4 for $\varepsilon = 8$.

The last two rows of Table 3 show accuracy gains for PDS CIFAR10 due to our aggregation methods. As before, **the highest accuracy gains are due to our** UTA_{tr}. Specifically, for ε of 1 and 8, the relative (absolute) accuracy gains due to UTA_{tr} are 16.72% (7.88%) and 30.11% (18.45%) over the DP-SGD baseline, and they are, respectively, 1.79% (0.97%) and 1.53% (1.2%) over $\text{EMA}_{\text{baseline}}$. Among the inference aggregations, OPA provides the maximum absolute accuracy gains over the DP-SGD baseline of 7.45% and 17.37%, respectively, for both $\varepsilon \in \{1, 8\}$. From the rightmost two plots (Figure 5), we see that DP-SGD baseline models exhibit very large variance with PDS CIFAR10 across training steps, but all the inference aggregation methods completely eliminate the variance.

Note that the improvements in PDS settings are significantly higher than that in the original settings, because the variance in model accuracy over training steps is large in PDS settings. Hence, the benefits of checkpoints aggregations magnify in these settings. For the PDS StackOverflow, where improvements are similar to StackOverflow, we hypothesize that this might be due to the distributions in PDS CIFAR10 (completely different images from even/odd classes) being significantly farther apart compared to the distributions in PDS StacktOverflow (text from questions/answers).

4.4 Experiments with Sample-level Privacy for CIFAR100 Dataset

In this section, we evaluate our aggregation methods (Section 4.2.1) in a *sample-level DP* setting with the original CIFAR100 and CIFAR100 with periodic distribution shifts (PDS).

4.4.1 Improving CIFAR100 baseline. First, we present a significant improvement over the SOTA baseline of [26], i.e., "No Agg" baseline in Table 4). In particular, *unlike in [26], we fine-tune the final EMA checkpoint*, i.e., the one computed using EMA during pre-training over ImageNet. This results in major accuracy boosts of 5% (70.3% → 75.51%) for $\varepsilon = 1$ and of 3.2% (77.6% → 80.81%) for $\varepsilon = 8$ for the original CIFAR100 task. We obtain similarly high improvements by fine-tuning the EMA of pre-trained checkpoints (instead of the final checkpoint) for the PDS-CIFAR100 case. We emphasize that these gains are *even before we use our aggregation methods*. We leave the further investigation of this phenomena to the future work.

4.4.2 Results for CIFAR100 and PDS CIFAR100. We first discuss the gains for original CIFAR100 due to our aggregation methods; Table 4 shows the results. We note significant performance gains for CIFAR100 due to almost all of our aggregation methods. **For both $\varepsilon \in \{1, 8\}$,** UTA_{tr} **provides the highest accuracy gains**: For ε of 1 and 8, the relative (absolute) accuracy gains due to UTA_{tr} are 0.89% (0.67%) and 0.91% (0.73%) over our improved DP-SGD baseline, and they are 1.4% (1.05%) and 0.82% (0.66%) over $\text{EMA}_{\text{baseline}}$. Among the inference aggregations, for $\varepsilon = 1$, UTA_{inf} provides the maximum

Table 4: Test accuracy gains for original and periodic distribution shifting (PDS) CIFAR100. We present techniques from prior works (DP-SGD and $\mathsf{EMA}_{\mathsf{baseline}}$ [26]) in blue.

DP	Baseline	Training Aggregations		Inference Aggregations				
(ε)	(No Agg)	$\mathsf{EMA}_{\mathsf{tr}}$	$\mathsf{UTA}_{\mathsf{tr}}$	$\mathsf{EMA}_{\mathsf{baseline}}$	$\mathsf{EMA}_{\mathsf{inf}}$	$\mathsf{UTA}_{\mathsf{inf}}$	OPA	OMV
CIFAR100; DP-SGD; sample-level privacy								
8	80.81 ± 0.11	81.23 ± 0.07	**81.54 ± 0.08**	80.88 ± 0.10	80.88 ± 0.10	80.83 ± 0.09	80.92 ± 0.10	80.82 ± 0.10
1	75.51 ± 0.19	75.58 ± 0.09	**76.18 ± 0.11**	75.13 ± 0.20	75.42 ± 0.13	75.62 ± 0.12	75.51 ± 0.16	75.57 ± 0.18
Periodic Distribution Shifting (PDS) CIFAR100; DP-SGD; sample-level privacy								
8	77.16 ± 0.11	79.83 ± 0.05	**81.27 ± 0.06**	80.53 ± 0.07	80.53 ± 0.07	80.53 ± 0.08	80.49 ± 0.08	80.41 ± 0.09
1	70.84 ± 0.16	74.88 ± 0.09	**75.81 ± 0.13**	74.61 ± 0.13	75.08 ± 0.12	75.81 ± 0.16	75.01 ± 0.17	74.97 ± 0.18

Table 5: *Relative* improvement in test *AUC-loss* compared to DPSGD (No Agg) baseline for proprietary pCVR Dataset. The two numbers presented for each algorithm are the improvements in the mean and standard deviation of the AUC-loss.

DP	Training Aggregations		Inference Aggregations				
(ε)	$\mathsf{EMA}_{\mathsf{tr}}$	$\mathsf{UTA}_{\mathsf{tr}}$	$\mathsf{EMA}_{\mathsf{baseline}}$	$\mathsf{EMA}_{\mathsf{inf}}$	$\mathsf{UTA}_{\mathsf{inf}}$	OPA	OMV
pCVR; DP-SGD; sample-level privacy; (mean, std)							
6	+0.32%, +18.9%	**+0.53%, +26.2%**	+0.19%, + 0.3%	+0.22%, +7%	+0.19%, +27.7%	**+0.54%, +62.6%**	N/A

relative (absolute) accuracy gain of 0.15% (0.11%), while for $\varepsilon = 8$, OPA provides the gain of 0.14% (0.11%) over our improved DP-SGD baseline. The gains for CIFAR100 are seemingly smaller than those for CIFAR10, but as mentioned in Section 1, CIFAR100 with 100 classes is a much more difficult task, and hence, the accuracy gains in DP regime are notable.

For **PDS CIFAR100 task as well,** $\mathsf{UTA}_{\mathsf{tr}}$ **provides the highest accuracy gains**: For ε of 1 and 8, the relative (absolute) accuracy gains due to $\mathsf{UTA}_{\mathsf{tr}}$ are 7.0% (4.97%) and 5.33% (4.11%) over our improved DP-SGD baseline, and they are 1.87% (1.4%) and 0.92% (0.74%) over $\mathsf{EMA}_{\mathsf{baseline}}$.

4.5 Experiments with Sample-level Privacy for pCVR

As this is a proprietary dataset, similar as prior works [25, 27], we report only the *relative* improvements in the AUC-loss; note that lower AUC-loss corresponds to better utility and improvement in AUC-loss means reduction in AUC-loss. The baseline we compare against is the model trained with DP-SGD ("No Agg"). The DP-SGD baseline has < 5% higher AUC-loss over the non-private model, which is similar to or slightly better than the DP-SGD models in prior work [25, 27]. Furthermore, as model stability is important for pCVR tasks, and DP training is well-known to increase variance, we also report the relative improvement in the standard deviation of the AUC-loss.

Table 5 presents the results. Similar to the other datasets, **all checkpoint aggregations improve AUC-loss**, i.e., reduce AUC-loss compared to the baseline. $\mathsf{EMA}_{\mathsf{tr}}$, $\mathsf{UTA}_{\mathsf{tr}}$, $\mathsf{UTA}_{\mathsf{inf}}$, $\mathsf{OPA}_{\mathsf{inf}}$ also reduce the variance significantly. Among all aggregation methods, $\mathsf{OPA}_{\mathsf{inf}}$ provides the largest (relative) improvements in AUC-loss and its standard deviation of 0.54% and 62.6%, respectively, over the DP-SGD baseline. Notice that in the context of ads ranking, even 0.1% relative improvement can have significant impact on revenue [87].

5 Quantifying uncertainty due to differential privacy noise

The prior literature on improving differentially private (DP) ML has focused on improving performances of DP models. However, a major issue with DP ML algorithms is high variance in their outputs due to high amounts of noise DP adds during training. High variance in outputs, i.e., DP ML models, reduces the confidence of these models in their predictions which is undesired in practical applications. Hence, quantifying uncertainty in outputs of DP ML algorithms is instrumental towards success of DP ML in practice.

Unfortunately, no prior work systematically investigates approaches for uncertainty quantification of DP deep learning. In this section, *we propose the first method to quantify the uncertainty that the DP noise adds to the outputs of DP ML algorithms, without additional privacy cost or computation.* In particular, we show that one can use the models along the path of DP-SGD to obtain an estimator for the variance introduced in the prediction due to the noise injected in the training process. We emphasize that the techniques in this section can be applied concurrently with the previous techniques in the paper. That is, in conjunction with the previous results, the results in this section imply a method to simultaneously improve utility and derive uncertainty estimates for a single training run at no additional cost.

For a bounded prediction function $f(\theta^{\mathrm{DP-SGD}})$ (with $\theta^{\mathrm{DP-SGD}}$ being the final model output by DP-SGD), a natural estimator of its variance is the "independent runs estimator:" running the algorithm independently k times to obtain $\left\{f\left(\theta_1^{\mathrm{DP-SGD}}\right), \ldots, f\left(\theta_k^{\mathrm{DP-SGD}}\right)\right\}$, and then obtaining the sample variance of this set of predictions [16]. However, the variance estimate is a post-processing of k runs of DP-SGD, which means roughly speaking both its privacy and computational cost are k times worse than DP-SGD. In particular, if we are restricted to one training of run of DP-SGD (e.g. due to computational costs), this method can only get one sample, i.e. the sample variance is undefined.

In this section, *we demonstrate a variance estimator that can give an estimate using only a single run of DP-SGD, and also can* *outperform the independent runs estimator in some settings even when more than a single run is allowed.*

5.1 Two Birds, One Stone: Our Uncertainty Estimator

To address the two hurdles discussed above, we propose a simple yet efficient method that leverages intermediate checkpoints computed during a single run of DP-SGD. Specifically, we substitute the k output models from the independent runs method with k checkpoints from a single run. The rest of the confidence interval computation remains the same for both the methods.

We first give a theoretical upper bound on the error between the sample variance of a statistic calculated at k intermediate checkpoints, and the true variance of this statistic at the final checkpoint. Our bias bound is decaying in two quantities: (i) the number of iterations t_1 before the first checkpoint, and (ii) γ, the minimum time between any two checkpoints. At a high level, our bound says that while checkpoints in DP-SGD are correlated, the addition of noise decreases their correlation over time, which justifies using them for uncertainty estimation in practice.

Our bound, proved in Section A.1, is as follows:

Theorem 5.1 (Simplified version of Theorem A.1). *Suppose $\mathcal{L}(\theta; D)$ is 1-strongly convex and M-smooth, and $\sigma = 1$ in DP-SGD. Let $0 < t_1 < t_2 < \ldots < t_k$ be such that $t_{i+1} \geq t_i + \gamma$ for $\forall i > 0$ and some minimum separation γ. Let $\{\theta_{t_i} : i \in [k]\}$ be the checkpoints, and $f : \Theta \rightarrow [-1, 1]$ be a statistic whose variance we wish to estimate. Let $V = \mathbf{Var}\left[f(\theta_{t_k})\right]$, i.e. the variance of statistic at the final checkpoint (i.e., the final model), $\mu = \frac{1}{k}\sum_{i=1}^{k} f(\theta_{t_i})$ be the sample mean, and $S = \left(\frac{1}{k-1}\sum_{i=1}^{k}(f(\theta_{t_i}) - \mu)^2\right)$ be the sample variance of the checkpoints. Then, for some "burn-in" times κ_1, κ_2 that are a function of θ_0, M, p, we have:*

$$|\mathbb{E}[S] - V| = \exp(-\Omega(\min\{t_1 - \kappa_1, \gamma - \kappa_2\})).$$

Here, the expectation $\mathbb{E}[\cdot]$ and the variance $\mathbf{Var}[\cdot]$ are over the randomness of DP-SGD.

5.1.1 *Proof Intuition.*

To simplify the proof in Section A.1 we actually prove a bound on the DP-LD algorithm, which is a continuous-time analog of DP-SGD. We defer a detailed discussion on the relationship between DP-LD and DP-SGD to Section A.1. For the following discussion, one should think of DP-LD and DP-SGD (with a small step size) as interchangeable.

Theorem 5.1 and its proof say the following: (i) As we increase t_1, the time before the first checkpoint, each of the checkpoints' marginal distributions approaches the distribution of θ_{t_k}, and (ii) As we increase γ, the time between checkpoints, the checkpoints' distributions approach pairwise independence. So increasing both t_1 and γ causes our checkpoints to approach k pairwise independent samples from the same distribution, i.e., our variance estimator approaches the true variance in expectation. To show both (i) and (ii), we build upon past results from the sampling literature to show a mixing bound of the following form: running DP-SGD from any point initialization θ_0, the Rényi divergence between θ_t and the limit as $t \rightarrow \infty$ of DP-LD, θ_∞, decays exponentially in t. This mixing bound shows (i) since if t_1 is sufficiently large, then the distributions of all of $\theta_{t_1}, \theta_{t_2}, \ldots, \theta_{t_k}$ are close to θ_∞, and thus close to each other. This also shows (ii) since DP-LD is a Markov chain, i.e. the distribution of θ_{t_j} conditioned on θ_{t_i} is equivalent to the distribution of $\theta_{t_j - t_i}$ if we run DP-LD starting from θ_{t_i} instead of θ_0. So our mixing bound shows that even after conditioning on θ_{t_i}, θ_{t_j} has distribution close to θ_∞. Since θ_{t_j} is close to θ_∞ conditioned on any value of θ_{t_i}, then θ_{t_j} is almost independent of θ_{t_i}.

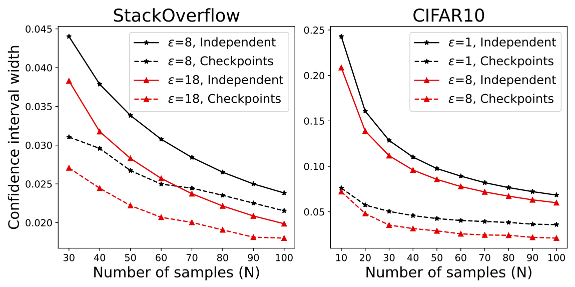

Figure 6: Uncertainty due to DP noise measured using confidence interval widths, computed via N bootstrap (independent) runs, and the last N checkpoints of a single run.

Remark: In Theorem 5.1, κ_1 is a function of θ_0 (the initialization model in DP-SGD) while κ_2 is independent of θ_0. In particular, κ_1 can be arbitrarily large compared to κ_2 if θ_0 is a poor choice for initialization, but we always have $\kappa_2 = O(\kappa_1)$. This implies the following:

- When the initialization is poor, using the sample variance of the checkpoints as an estimator gives *a computational improvement* over the sample variance of k independent runs of a training algorithm.
- Regardless of the initialization, using the sample variance of k checkpoints is *never worse* in terms of computation cost than using k independent runs.
- Checkpoints can provide tighter confidence intervals than independent runs under a fixed privacy constraint: Suppose we have a fixed noise multiplier $\sigma/(L/n)$ we would like to use in training, as well as a fixed privacy budget. This implies we have a fixed number of iterations T we can run. Fix t_1 and γ such that the sample variance of the checkpoints has low bias; since κ_1 can be much larger than κ_2, we should also set t_1 to be much larger than t_2. Suppose we want to construct a confidence interval for a model trained for at least t_1 iterations. Using independent runs, we can get T/t_1 samples. Using checkpoints from one T-iteration run, we can get $1 + \frac{T-t_1}{\gamma}$ samples. So we can get $\approx t_1/\gamma$ times as many samples by using checkpoints, and thus get a narrower confidence interval under the same privacy budget.

5.1.2 *Empirical Analysis on Quadratic Losses.*

We perform an empirical study of using the checkpoint variance estimator. We consider running DP-SGD on a 1-dimensional quadratic loss; we ignore clipping for simplicity, and assume the training rounds/privacy budget are fixed such that we can do exactly 128 rounds of DP-SGD. We set the learning rate $\eta = .07$, set the Gaussian variance such that the distribution of the final iterate has variance exactly 1, and set the initialization to be a random point drawn from $\mathcal{N}(0, \sigma^2 = 100^2)$. Since $(1-\eta)^{64} \approx 1/100$, under these parameters it takes roughly 64 rounds

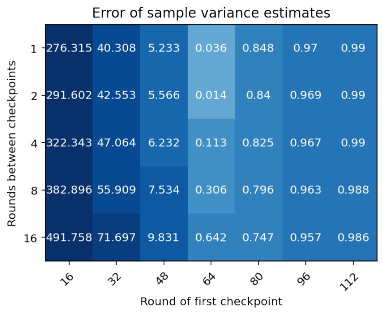

Figure 7: RMSE of the average sample variance given by the checkpoint estimator on quadratic losses.

for DP-SGD to converge to within distance 1 of the minimizer. This reflects the setting where the burn-in time is a significant fraction of the training time, i.e. where Lemma 5.1 offers improvements over independent runs. We vary the burn-in time (i.e. round number of the first checkpoint) and the number of rounds between each checkpoint (i.e., the total number of checkpoints used) used in the variance estimator, and compute the error of the variance estimator across 1000 runs.

In Figure 7 we plot the RMSE of the variance estimator, which accounts for both the bias and variance of the estimator (note that Lemma 5.1 only looks at the bias; in Section A.2 we discuss the problem of optimizing the checkpoints to minimize the RMSE). As predicted by Lemma 5.1, we see that using too small a burn-in time causes a large bias, as the DP-SGD process has not had time to converge before the first checkpoint. We also see that using too large a burn-in time is suboptimal, since it reduces the number of checkpoints available to use in the estimator, increasing its variance. For rounds between checkpoints, at the best burn-in time of 64, we see it is best to choose 2 rounds between checkpoints. Again this matches the intuition of Lemma 5.1: if we choose 1 round between checkpoints, checkpoints become too correlated which introduces bias into the variance estimate. At the same time, if we choose a larger separation like 16, we reduce the number of checkpoints the estimator uses, which increases the estimator's variance.

Recall that using independent runs of 128 iterations the independent runs' variance estimate is undefined, so all results in Figure 7 are improvements over that method. Even with e.g. 2 independent runs of 64 iterations, we only get 2 samples. Ignoring the bias due to using fewer iterations, the variance of this estimator is the variance of a degree-1 chi-squared distribution which is 2, i.e. it achieves RMSE at least 2.

5.1.3 Empirical Analysis on Deep Learning. We compare the uncertainty quantified using the independent runs method and using our method; experimental setup is the same as in Section 4. First, for a given dataset, we do 101 independent training runs (no budget split). For accurately measuring the uncertainty of the training run at the specified privacy budget, we do not split the privacy budget across the independent runs here. Note that this is a superior baseline, as the overall privacy budget is significantly increased. To compute uncertainty using the independent runs method for a fixed N, we first take the final model from N of these runs (chosen randomly). Given an input sample, we compute prediction scores for each model, and compute the 95% confidence interval width for the highest mean score. We compute the average of the confidence interval widths in this manner for every sample from the validation set[3]. We conduct five independent repeats of this method, and report the mean confidence interval width as our final uncertainty estimate. For computing uncertainty using our checkpoints based method, we do not optimize for the separation between checkpoints, giving a weaker hyperparameter-free method. we instead select the last N checkpoints (i.e., last N iterations) from a random training run, and obtain average confidence interval widths as above. T

Figure 6 shows the results for StackOverflow and CIFAR10. We see that the widths computed using intermediate checkpoints consistently gives a reasonable lower bound on the widths computed using independent runs, despite the strong baseline optimizing for the separation between checkpoints. For instance, for DP-FTRL training on StackOverflow, the confidence widths due to independent runs are always within a factor of 2 of the widths provided by our method across various privacy levels; for DP-SGD on CIFAR10, the bound is a factor is 4.

6 Conclusions

In this work, we design a general adaptive checkpoint aggregation framework to increase the performances of state-of-the-art DP ML techniques. We show that uniform tail averaging of improves the excess empirical risk bound compared to the last checkpoint of DP-SGD. We demonstrate that uniform tail averaging *during training* can provide significant improvements in prediction performances over the state-of-the-art for CIFAR10 and StackOverflow datasets, and the gains get magnified in more real-world settings with periodically varying training data distributions. Lastly, we prove that for some standard loss functions, the sample variance from last few checkpoints provides a good approximation of the variance of the final model of a DP run. Empirically, we show that the last few checkpoints can provide a reasonable lower bound for the variance of a converged DP model.

Acknowledgments

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* (2016).

[2] Martín Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM.

[3] Martín Abadi, Andy Chu, Ian J. Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep Learning with Differential Privacy. In *Proc. of the 2016 ACM SIGSAC Conf. on Computer and Communications Security (CCS'16)*. 308–318.

[4] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U Rajendra Acharya, et al. 2021. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion* 76 (2021), 243–297.

[5] Ehsan Amid, Arun Ganesh, Rajiv Mathews, Swaroop Ramaswamy, Shuang Song, Thomas Steinke, Vinith M Suriyakumar, Om Thakkar, and Abhradeep Thakurta.

[3]Due to the large size of StackOverflow test data, we instead use validation data.

2022. Public data-assisted mirror descent for private model training. In *International Conference on Machine Learning*. PMLR, 517–535.

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

[7] Giuseppe Ateniese, Giovanni Felici, Luigi V Mancini, Angelo Spognardi, Antonio Villani, and Domenico Vitali. 2013. Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. *arXiv preprint arXiv:1306.4447* (2013).

[8] Giuseppe Ateniese, Luigi V Mancini, Angelo Spognardi, Antonio Villani, Domenico Vitali, and Giovanni Felici. 2015. Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. *International Journal of Security and Networks* 10, 3 (2015), 137–150.

[9] Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, Trevor Cai, Aidan Clark, Ivo Danihelka, Claudio Fantacci, Jonathan Godwin, Chris Jones, Ross Hemsley, Tom Hennigan, Matteo Hessel, Shaobo Hou, Steven Kapturowski, Thomas Keck, Iurii Kemaev, Michael King, Markus Kunesch, Lena Martens, Hamza Merzic, Vladimir Mikulik, Tamara Norman, John Quan, George Papamakarios, Roman Ring, Francisco Ruiz, Alvaro Sanchez, Rosalia Schneider, Eren Sezener, Stephen Spencer, Srivatsan Srinivasan, Luyu Wang, Wojciech Stokowiec, and Fabio Viola. 2020. *The DeepMind JAX Ecosystem*. http://github.com/deepmind

[10] Borja Balle, Peter Kairouz, Brendan McMahan, Om Thakkar, and Abhradeep Guha Thakurta. 2020. Privacy amplification via random check-ins. *Advances in Neural Information Processing Systems* 33 (2020), 4623–4634.

[11] Andrés F. Barrientos, Jerome P. Reiter, Ashwin Machanavajjhala, and Yan Chen. 2019. Differentially Private Significance Tests for Regression Coefficients. *Journal of Computational and Graphical Statistics* 28, 2 (2019), 440–453. https://doi.org/10.1080/10618600.2018.1538881 arXiv:https://doi.org/10.1080/10618600.2018.1538881

[12] Raef Bassily, Adam Smith, and Abhradeep Thakurta. 2014. 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)*. 464–473.

[13] Raef Bassily, Adam Smith, and Abhradeep Thakurta. 2014. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*. IEEE.

[14] Edmon Begoli, Tanmoy Bhattacharya, and Dimitri Kusnezov. 2019. The need for uncertainty quantification in machine-assisted medical decision making. *Nature Machine Intelligence* 1, 1 (2019), 20–23.

[15] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, and Skye Wanderman-Milne. 2018. JAX: composable transformations of Python+NumPy programs. http://github.com/google/jax

[16] Thomas Brawner and James Honaker. 2018. Bootstrap inference and differential privacy: Standard errors for free. *Unpublished Manuscript* (2018).

[17] Andy Brock, Soham De, Samuel L. Smith, and Karen Simonyan. 2021. High-Performance Large-Scale Image Recognition Without Normalization. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 1059–1071. http://proceedings.mlr.press/v139/brock21a.html

[18] Gavin R Brown, Krishnamurthy Dj Dvijotham, Georgina Evans, Daogao Liu, Adam Smith, and Abhradeep Guha Thakurta. 2024. Private Gradient Descent for Linear Regression: Tighter Error Bounds and Instance-Specific Uncertainty Estimation. In *Proceedings of the 41st International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 235)*, Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (Eds.). PMLR, 4561–4584. https://proceedings.mlr.press/v235/brown24a.html

[19] Mark Bun and Thomas Steinke. 2016. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography Conference*. Springer, 635–658.

[20] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. 2022. Membership inference attacks from first principles. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1897–1914.

[21] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. 2019. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*. 267–284.

[22] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. 2021. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*. 2633–2650.

[23] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. 2011. Differentially private empirical risk minimization. *Journal of Machine Learning Research* 12, Mar (2011), 1069–1109.

[24] Rishav Chourasia, Jiayuan Ye, and Reza Shokri. 2021. Differential privacy dynamics of langevin diffusion and noisy gradient descent. *Advances in Neural Information Processing Systems* 34 (2021), 14771–14781.

[25] Lynn Chua, Qiliang Cui, Badih Ghazi, Charlie Harrison, Pritish Kamath, Walid Krichene, Ravi Kumar, Pasin Manurangsi, Krishna Giri Narra, Amer Sinha, et al. 2024. Training differentially private ad prediction models with semi-sensitive features. *arXiv preprint arXiv:2401.15246* (2024).

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

[27] Carson Denison, Badih Ghazi, Pritish Kamath, Ravi Kumar, Pasin Manurangsi, Krishna Giri Narra, Amer Sinha, Avinash V Varadarajan, and Chiyuan Zhang. 2022. Private ad modeling with DP-SGD. *arXiv preprint arXiv:2211.11896* (2022).

[28] Sergey Denisov, Brendan McMahan, Keith Rush, Adam Smith, and Abhradeep Guha Thakurta. 2022. Improved Differential Privacy for SGD via Optimal Private Linear Operators on Adaptive Streams. *arXiv preprint arXiv:2202.08312* (2022).

[29] Cynthia Dwork. 2008. Differential privacy: A survey of results. In *International Conference on Theory and Applications of Models of Computation*. 1–19.

[30] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *Proc. of the Third Conf. on Theory of Cryptography (TCC)*. 265–284. http://dx.doi.org/10.1007/11681878_14

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

[32] Murat A. Erdogdu, Rasa Hosseinzadeh, and Matthew Shunshi Zhang. 2020. Convergence of Langevin Monte Carlo in Chi-Squared and Rényi Divergence. In *International Conference on Artificial Intelligence and Statistics*.

[33] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Shuang Song, Kunal Talwar, and Abhradeep Thakurta. 2020. Encode, Shuffle, Analyze Privacy Revisited: Formalizations and Empirical Evaluation. *CoRR* abs/2001.03618 (2020). arXiv:2001.03618 https://arxiv.org/abs/2001.03618

[34] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. 2019. Amplification by Shuffling: From Local to Central Differential Privacy via Anonymity. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, Timothy M. Chan (Ed.). SIAM, 2468–2479. https://doi.org/10.1137/1.9781611975482.151

[35] Georgina Evans, Gary King, Margaret Schwenzfeier, and Abhradeep Thakurta. 2020. Statistically Valid Inferences from Privacy Protected Data. *American Political Science Review* (2020).

[36] Vitaly Feldman, Tomer Koren, and Kunal Talwar. 2020. Private Stochastic Convex Optimization: Optimal Rates in Linear Time. In *Proc. of the Fifty-Second ACM Symp. on Theory of Computing (STOC'20)*.

[37] Vitaly Feldman, Audra McMillan, and Kunal Talwar. 2022. Hiding among the clones: A simple and nearly optimal analysis of privacy amplification by shuffling. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 954–964.

[38] Vitaly Feldman, Ilya Mironov, Kunal Talwar, and Abhradeep Thakurta. 2018. Privacy amplification by iteration. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 521–532.

[39] Cecilia Ferrando, Shufan Wang, and Daniel Sheldon. 2022. Parametric Bootstrap for Differentially Private Confidence Intervals. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 1598–1618.

[40] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM.

[41] Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. 2014. Privacy in Pharmacogenetics: An End-to-End Case Study of Personalized Warfarin Dosing.. In *USENIX Security Symposium*.

[42] Arun Ganesh and Kunal Talwar. 2020. Faster Differentially Private Samplers via Rényi Divergence Analysis of Discretized Langevin MCMC. *CoRR* abs/2010.14658 (2020). https://arxiv.org/abs/2010.14658

[43] Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, Muhammad Shahzad, Wen Yang, Richard Bamler, and Xiao Xiang Zhu. 2023. A survey of uncertainty in deep neural networks. *Artif. Intell. Rev.* 56, Suppl 1 (July 2023), 1513–1589. https://doi.org/10.1007/s10462-023-10562-9

[44] Nicholas J. A. Harvey, Christopher Liaw, Yaniv Plan, and Sikander Randhawa. 2019. Tight Analyses for Non-Smooth Stochastic Gradient Descent. In *COLT*.

[45] Christian Hubschneider, Robin Hutmacher, and J Marius Zöllner. 2019. Calibrating uncertainty models for steering angle estimation. In *2019 IEEE intelligent transportation systems conference (ITSC)*. IEEE, 1511–1518.

[46] Prateek Jain, Dheeraj M. Nagaraj, and Praneeth Netrapalli. 2021. Making the Last Iterate of SGD Information Theoretically Optimal. *SIAM Journal on Optimization* 31, 2 (2021), 1108–1130. https://doi.org/10.1137/19M128908X arXiv:https://doi.org/10.1137/19M128908X

[47] Kaggle. 2018. The StackOverflow data. https://www.kaggle.com/datasets/stackoverflow/stackoverflow. [Online; accessed 15-September-2022].
[48] Peter Kairouz, Brendan McMahan, Shuang Song, Om Thakkar, Abhradeep Thakurta, and Zheng Xu. 2021. Practical and private (deep) learning without sampling or shuffling. In *International Conference on Machine Learning*. PMLR, 5213–5225.
[49] Vishesh Karwa and Salil Vadhan. 2017. Finite sample differentially private confidence intervals. *arXiv preprint arXiv:1711.03908* (2017).
[50] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).
[51] Haoran Li, Li Xiong, Lucila Ohno-Machado, and Xiaoqian Jiang. 2014. Privacy preserving RBF kernel support vector machine. *BioMed Research International* (2014).
[52] Patrick L McDermott and Christopher K Wikle. 2019. Deep echo state networks with uncertainty quantification for spatio-temporal forecasting. *Environmetrics* 30, 3 (2019), e2553.
[53] Brendan McMahan, Abhradeep Thakurta, Galen Andrew, Borja Balle, Peter Kairouz, Daniel Ramage, Shuang Song, Thomas Steinke, Andreas Terzis, Om Thakkar, and Zheng Xu. 2022. Federated learning with formal differential privacy guarantees. https://ai.googleblog.com/2022/02/federated-learning-with-formal.html. [Online; accessed 15-September-2022].
[54] H Brendan McMahan, Galen Andrew, Ulfar Erlingsson, Steve Chien, Ilya Mironov, Nicolas Papernot, and Peter Kairouz. 2018. A general approach to adding differential privacy to iterative training procedures. *arXiv preprint arXiv:1812.06210* (2018).
[55] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. *Proceedings of the 20th AISTATS* (2017).
[56] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2017. Learning differentially private recurrent language models. *arXiv preprint arXiv:1710.06963* (2017).
[57] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. 2019. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE symposium on security and privacy (SP)*. IEEE, 691–706.
[58] Ilya Mironov. 2017. Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*. IEEE, 263–275.
[59] Tom M Mitchell. 1980. *The need for biases in learning generalizations*. Department of Computer Science, Laboratory for Computer Science Research ....
[60] Tanya Nair, Doina Precup, Douglas L Arnold, and Tal Arbel. 2020. Exploring uncertainty measures in deep networks for multiple sclerosis lesion detection and segmentation. *Medical image analysis* 59 (2020), 101557.
[61] Milad Nasr, Reza Shokri, and Amir Houmansadr. 2018. Machine learning with membership privacy using adversarial regularization. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*. 634–646.
[62] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2007. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*. 75–84.
[63] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. 2019. Knockoff nets: Stealing functionality of black-box models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4954–4963.
[64] Nicolas Papernot and Thomas Steinke. 2022. Hyperparameter tuning with renyi differential privacy. *ICLR* (2022).
[65] Nicolas Papernot, Abhradeep Thakurta, Shuang Song, Steve Chien, and Úlfar Erlingsson. 2020. Tempered sigmoid activations for deep learning with differential privacy. *arXiv preprint arXiv:2007.14191* (2020).
[66] Stephan Rabanser, Anvith Thudi, Abhradeep Guha Thakurta, Krishnamurthy Dj Dvijotham, and Nicolas Papernot. 2023. Training Private Models That Know What They Don't Know. In *Thirty-seventh Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=EgCjf1vjMB
[67] Swaroop Ramaswamy, Om Thakkar, Rajiv Mathews, Galen Andrew, H Brendan McMahan, and Françoise Beaufays. 2020. Training production language models without memorizing user data. *arXiv preprint arXiv:2009.10031* (2020).
[68] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and H Brendan McMahan. 2020. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295* (2020).
[69] Abhijit Guha Roy, Sailesh Conjeti, Nassir Navab, and Christian Wachinger. 2018. Inherent brain segmentation quality control from fully convnet monte carlo sampling. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 664–672.
[70] Théo Ryffel, Francis Bach, and David Pointcheval. 2022. Differential Privacy Guarantees for Stochastic Gradient Langevin Dynamics. *arXiv preprint arXiv:2201.11980* (2022).
[71] Sriram Sankararaman, Guillaume Obozinski, Michael I Jordan, and Eran Halperin. 2009. Genomic privacy and limits of individual detection in a pool. *Nature genetics* 41, 9 (2009), 965–967.
[72] Ohad Shamir and Tong Zhang. 2013. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *International Conference on Machine Learning*. 71–79.
[73] Virat Shejwalkar and Amir Houmansadr. 2021. Membership privacy for machine learning models through knowledge transfer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 9549–9557.
[74] Virat Shejwalkar, Huseyin A Inan, Amir Houmansadr, and Robert Sim. 2021. Membership inference attacks against nlp classification models. In *NeurIPS 2021 Workshop Privacy in Machine Learning*.
[75] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*. 3–18.
[76] Congzheng Song and Vitaly Shmatikov. 2019. Overlearning Reveals Sensitive Attributes. In *International Conference on Learning Representations*.
[77] Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. 2013. Stochastic gradient descent with differentially private updates. In *2013 IEEE Global Conference on Signal and Information Processing*. IEEE, 245–248.
[78] Shuang Song, Thomas Steinke, Om Thakkar, and Abhradeep Thakurta. 2021. Evading the Curse of Dimensionality in Unconstrained Private GLMs. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 130)*, Arindam Banerjee and Kenji Fukumizu (Eds.). PMLR, 2638–2646. https://proceedings.mlr.press/v130/song21a.html
[79] Natasa Tagasovska and David Lopez-Paz. 2019. Single-model uncertainties for deep learning. *Advances in Neural Information Processing Systems* 32 (2019).
[80] Mingxing Tan and Quoc V. Le. 2019. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 6105–6114. http://proceedings.mlr.press/v97/tan19a.html
[81] Xinyu Tang, Saeed Mahloujifar, Liwei Song, Virat Shejwalkar, Milad Nasr, Amir Houmansadr, and Prateek Mittal. 2022. Mitigating Membership Inference Attacks by {Self-Distillation} Through a Novel Ensemble Architecture. In *31st USENIX Security Symposium (USENIX Security 22)*. 1433–1450.
[82] Florian Tramer and Dan Boneh. 2020. Differentially Private Learning Needs Better Features (or Much More Data). In *International Conference on Learning Representations*.
[83] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. 2016. Stealing machine learning models via prediction apis. In *USENIX Security*.
[84] Tim van Erven and Peter Harremos. 2014. Rényi Divergence and Kullback-Leibler Divergence. *IEEE Transactions on Information Theory* 60, 7 (2014), 3797–3820. https://doi.org/10.1109/TIT.2014.2320500
[85] Santosh Vempala and Andre Wibisono. 2019. Rapid Convergence of the Unadjusted Langevin Algorithm: Isoperimetry Suffices. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/65a99bb7a3115fdede20da98b08a370f-Paper.pdf
[86] Guotai Wang, Wenqi Li, Michael Aertsen, Jan Deprest, Sébastien Ourselin, and Tom Vercauteren. 2019. Aleatoric uncertainty estimation with test-time augmentation for medical image segmentation with convolutional neural networks. *Neurocomputing* 338 (2019), 34–45.
[87] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*. 1–7.
[88] Yu-Xiang Wang, Borja Balle, and Shiva Prasad Kasiviswanathan. 2019. Subsampled Renyi Differential Privacy and Analytical Moments Accountant. In *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan*. 1226–1235.
[89] Max Welling and Yee W Teh. 2011. Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*. 681–688.
[90] Qiyiwen Zhang, Zhiqi Bu, Kan Chen, and Qi Long. 2021. Differentially Private Bayesian Neural Networks on Accuracy, Privacy and Reliability. https://doi.org/10.48550/ARXIV.2107.08461
[91] Yuchen Zhang, Percy Liang, and Moses Charikar. 2017. A hitting time analysis of stochastic gradient langevin dynamics. In *Conference on Learning Theory*. PMLR, 1980–2022.
[92] Zuhe Zhang, Benjamin IP Rubinstein, and Christos Dimitrakakis. 2016. On the differential privacy of Bayesian inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 30.
[93] Chen Zhu, Zheng Xu, Mingqing Chen, Jakub Konečnỳ, Andrew Hard, and Tom Goldstein. 2021. Diurnal or Nocturnal? Federated Learning of Multi-branch Networks from Periodically Shifting Distributions. In *International Conference on Learning Representations*.
[94] Yuqing Zhu and Yu-Xiang Wang. 2019. Poission subsampled rényi differential privacy. In *International Conference on Machine Learning*. PMLR, 7634–7642.

A Details and Extensions for Theorem 5.1

A.1 Proof of Theorem 5.1

For completeness, we review the formal setup for the theorem we wish to prove. We focus on DP-LD, defined as follows:

$$d\theta_t = -\nabla\mathcal{L}(\theta_t; D)dt + \sigma\sqrt{2}dW_t. \tag{9}$$

One can view DP-LD and DP-SGD as approximations of each other as follows. We first reformulate (unconstrained) DP-SGD with step size η as:

$$\widetilde{\theta}_{(t+1)\eta} \leftarrow \widetilde{\theta}_{t\eta} - \eta\nabla\mathcal{L}(\widetilde{\theta}_{t\eta}; D) + b_t, b_t \sim \mathcal{N}(0, 2\eta\sigma^2\mathbb{I}_{p\times p}).$$

This reparameterization is commonly known as (DP-)SGLD [24, 70, 89, 91]. Notice that we have reparameterized $\widetilde{\theta}$ so that its subscript refers to the sum of all step-sizes so far, i.e. after t iterations we have $\widetilde{\theta}_{t\eta}$ and not $\widetilde{\theta}_t$. Also notice that the variance of the noise we added is proportional to the step size η. In turn, for any η that divides t, after t/η iterations with step size η, the sum of variances of noises added is $2t\sigma^2$. This can be used to show a Renyi-DP guarantee for DP-SGLD with fixed t that is independent of η, including in the limit as $\eta \rightarrow 0$.

Now, taking the limit as η goes to 0 of the sequence of random variables $\{\widetilde{\theta}_{t\eta}\}_{t\in\mathbb{Z}_{\geq 0}}$ defined by DP-SGLD, we get a continuous sequence $\{\theta_t\}_{t\in\mathbb{R}_{\geq 0}}$. In particular, if we fix some t, then θ_t is the limit as η goes to 0 of $\widetilde{\theta}_t$ defined by DP-SGLD with step size η. This sequence is exactly the sequence defined by DP-LD.

Note that the solutions θ_t to this equation are random variables. A key property of DP-LD is that the stationary distribution (equivalently, the limiting distribution as $t \rightarrow \infty$) has pdf proportional to $\exp(-\mathcal{L}(\theta; D)/\sigma)$ under mild assumptions on $\mathcal{L}(\theta; D)$ (which are satisfied by strongly convex and smooth functions).

While we focus on DP-LD for simplicity of presentation, a similar result can be proven for DP-SGLD. We discuss this in Section A.4.

To simplify proofs and presentation in the section, we will assume that (a) θ_0 is a point distribution, (b) we are looking at unconstrained optimization over $\mathbb{R}^p$, i.e., there is no need for a projection operator in DP-SGD and DP-LD, (c) the loss $\mathcal{L}$ is 1-strongly convex and M-smooth, and (d) $\sigma = 1$. We note that (a) can be replaced with θ_0 being sampled from a random initialization without too much work, and (c) can be enforced for Lipschitz, smooth functions by adding a quadratic regularizer. We let θ^* refer to the (unique) minimizer of $\mathcal{L}$ throughout the section.

Now, we consider the following setup: We obtain a single sample of the trajectory $\{\theta_t : t \in [0, T]\}$. We have some statistic $f : \Theta \rightarrow [-1, 1]$, and we wish to estimate the variance of some weighted average of the statistic across the checkpoints at times $0 < t_1 < t_2 < t_3 < \ldots < t_k = T$, i.e. the variance $V := \mathbf{Var}\left(\sum_i p_i f(\theta_{t_i})\right)$, where $\sum_i p_i = 1, p_i \geq 0$. To do so, we use a rescaling of the sample variance of the checkpoints. That is, our estimator is defined as $S = \frac{\sum_{i=1}^k p_i^2}{k-1}\sum_{i=1}^k (f(\theta_{t_i}) - \widehat{\mu})^2$ where $\widehat{\mu} = \frac{1}{k}\sum_{i=1}^k f(\theta_{t_i})$.

Theorem A.1. *Under the preceding assumptions/setup, for some sufficiently large constant c, let*

$$\kappa_1 = \frac{1}{2M} + \ln(cM(\|\theta_0 - \theta^*\|_2^2 + p\ln(M))) + c\ln(1/\Delta),$$

$$\kappa_2 = \frac{1}{2M} + \ln(cM(\ln(1/\Delta) + p\ln(M))) + c\ln(1/\Delta),$$

(recall that p is the dimensionality of the space). Then, if $t_1 > \kappa_1$ and $t_{i+1} > t_i + \kappa_2$ for all $i > 0$, for S, V as defined above:

$$|\mathbb{E}[S] - V| = O(\Delta\sum_{i=1}^k p_i^2).$$

Theorem A.7 is the special case of setting $p_k = 1$ and $p_i = 0, i \neq k$. Note that κ_1 can be arbitrarily large compared to κ_2 due to its dependence on θ_0, whereas $\kappa_2 = O(\kappa_1)$. In particular, $\kappa_1 + (k-1)\kappa_2$ (the time to do one long run and use k intermediate checkpoints for uncertainty estimation) can be significantly smaller than $k\kappa_1$ (the time to do k independent runs and use the final checkpoints for uncertainty estimation). Before proving this theorem, we need a few helper lemmas about Rényi divergences:

Definition A.2. *The Rényi divergence of order $\alpha > 1$ between two distributions $\mathcal{P}$ and $\mathcal{Q}$ (with support $\mathbb{R}^d$), $D_\alpha(\mathcal{P}||\mathcal{Q})$, is defined as follows:*

$$D_\alpha(\mathcal{P}||\mathcal{Q}) := \int_{\theta\in\mathbb{R}^d}\frac{P(\theta)^\alpha}{Q(\theta)^{\alpha-1}}d\theta$$

We refer the reader to e.g. [58, 84] for properties of the Rényi divergence. The following property shows that for any two random variables close in Rényi divergence, functions of them are close in expectation:

Lemma A.3. *[Adapted from Lemma C.2 of [19]] Let $\mathcal{P}$ and $\mathcal{Q}$ be two distributions on Ω and $g : \Omega \rightarrow [-1, 1]$. Then,*

$$|\mathbb{E}_{x\sim\mathcal{P}}[g(x)] - \mathbb{E}_{x\sim\mathcal{Q}}[g(x)]| \leq \sqrt{e^{D_2(\mathcal{P}||\mathcal{Q})} - 1}.$$

Here, $D_2(\mathcal{P}||\mathcal{Q})$ corresponds to Rényi divergence of order two between the distributions $\mathcal{P}$ and $\mathcal{Q}$.

The next lemma shows that the solution to DP-LD approaches θ_∞ exponentially quickly in Rényi divergence.

Lemma A.4. *Fix some point θ_0. Assume $\mathcal{L}$ is 1-strongly convex, and M-smooth. Let $\mathcal{P}$ be the distribution of θ_t according to DP-LD for $\sigma = 1$ and:*

$$t := 1/2M + \ln(c(M\|\theta_0 - \theta^*\|_2^2 + p\ln(M))) + c\ln(1/\Delta).$$

Where c is a sufficiently large constant. Let Q be the stationary distribution of DP-LD. Then:

$$D_2(\mathcal{P}||\mathcal{Q}) = O(\Delta^2).$$

The proof of this lemma builds upon techniques in [42], and we defer it to the appendix. Our final helper lemma shows that θ_∞ is close to θ^* with high probability:

Lemma A.5. *Let θ_∞ be the random variable given by the stationary distribution of DP-LD for $\sigma = 1$. If $\mathcal{L}$ is 1-strongly convex, then:*

$$\mathbf{Pr}[\|\theta_\infty - \theta^*\|_2 > \sqrt{p} + x] \leq \exp(-x^2/2).$$

Proof. We know the stationary distribution has pdf proportional to $\exp(-\mathcal{L}(\theta_t; D))$. In particular, since $\mathcal{L}$ is 1-strongly convex, this means θ_∞ is a sub-Gaussian random vector (i.e., its dot product with any unit vector is a sub-Gaussian random variable), and thus the above tail bound applies to it. □

We now will show that under the assumptions in Theorem A.1, every checkpoint is close to the stationary distribution, and that every pair of checkpoints is nearly pairwise independent.

Lemma A.6. *Under the assumptions/setup of Theorem A.1, we have:*

(E1) $\forall i : |\mathbb{E}[(f(\theta_{t_i}))] - \mathbb{E}[(f(\theta_{t_k}))]| = O(\Delta)$,
(E2) $\forall i : |\mathbb{E}[(f(\theta_{t_i})^2)] - \mathbb{E}[f(\theta_{t_k})^2]| = O(\Delta)$,
(E3) $\forall i < j : |\mathbf{Cov}\left(f(\theta_{t_i}), f(\theta_{t_j})\right)| = O(\Delta)$.

Proof. We assume without loss of generality Δ is at most a sufficiently small constant; otherwise, since f has range $[-1, 1]$, all of the above quantities can easily be bounded by 2, so a bound of $O(\Delta)$ holds for any distributions on $\{\theta_{t_i}\}$.

For (E1), by triangle inequality, it suffices to prove a bound of $O(\Delta)$ on $|\mathbb{E}[f(\theta_{t_i})] - \mathbb{E}[f(\theta_\infty)]|$. We abuse notation by letting θ_t denote both the random variable and its distribution. Then:

$$\begin{aligned}
&|\mathbb{E}[f(\theta_{t_i})] - \mathbb{E}[f(\theta_\infty)]| \\
&\overset{\text{Lemma A.3}}{\leq} \sqrt{e^{D_2(f(\theta_{t_i}), f(\theta_\infty))} - 1} \overset{(*_1)}{\leq} \sqrt{e^{D_2(\theta_{t_i}, \theta_\infty)} - 1} \\
&\overset{\text{Lemma A.4}, t_i \geq \kappa_1}{=} \sqrt{e^{O(\Delta^2)} - 1} \overset{(*_2)}{=} O(\Delta).
\end{aligned}$$

In $(*_1)$ we use the data-processing inequality (Theorem 9 of [84]), and in $(*_2)$ we use the fact $e^x - 1 \leq 2x, x \in [0, 1]$ and our assumption on Δ.

(E2) follows from (E1) by just using f^2 (which is still bounded in $[-1, 1]$) instead of f.

For (E3), note that since DP-LD is a (continuous) Markov chain, the distribution of θ_{t_j} conditioned on θ_{t_i} is the same as the distribution of $\theta_{t_j - t_i}$ according to DP-LD if we start from θ_{t_i} instead of θ_0. Let $\mathcal{P}$ be the joint distribution of $\theta_{t_i}, \theta_{t_j}$. Let $\mathcal{Q}$ be the joint distribution of $\theta_{t_i}, \theta_\infty$ (since DP-LD has the same stationary distribution regardless of its initialization, this is a pair of independent variables). Let $\mathcal{P}', \mathcal{Q}'$ be defined identically to $\mathcal{P}||\mathcal{Q}$, except when sampling θ_{t_i}, if $\left\|\theta_{t_i} - \theta^*\right\|_2 > \sqrt{p} + \sqrt{2\ln(1/\Delta)}$ we instead set $\theta_{t_i} = \theta^*$ (and in the case of $\mathcal{P}'$, we instead sample θ_{t_j} from $\theta_{t_j} | \theta_{t_i} = \theta^*$ when this happens). Let $\mathcal{R}$ denote this distribution over θ_{t_i}. Then similarly to the proof of (E1) we have:

$$\begin{aligned}
&|\mathbb{E}_{\mathcal{P}'}[f(\theta_{t_i})f(\theta_{t_j})] - \mathbb{E}_{\mathcal{Q}'}[f(\theta_{t_i})]\mathbb{E}[f(\theta_\infty)]| \\
&\overset{\text{Lemma A.3}}{\leq} \sqrt{e^{D_2(\mathcal{P}', \mathcal{Q}')} - 1} \\
&\overset{(*_3)}{\leq} \sqrt{e^{\max_{\theta_{t_i} \in \text{supp}(\mathcal{R})} \{D_2(\theta_{t_j} | \theta_{t_i}, \theta_\infty)\}} - 1}. \\
&\overset{\text{Lemma A.4}, t_j - t_i \geq \kappa_2}{=} \sqrt{e^{O(\Delta^2)} - 1} = O(\Delta).
\end{aligned}$$

Here $(*_3)$ follows from the convexity of Rényi divergence, and in our application of A.4, we are using the fact that for all $\theta_{t_i} \in \text{supp}(\mathcal{R})$, $\left\|\theta_{t_i} - \theta^*\right\|_2 \leq \sqrt{p} + \sqrt{2\ln(1/\Delta)}$. Furthermore, by Lemma A.5, we know $\mathcal{P}$ and $\mathcal{P}'$ (resp. $\mathcal{Q}$ and $\mathcal{Q}'$) differ by at most Δ in total variation distance. So, since f is bounded in $[-1, 1]$, we have:

$$\begin{aligned}
|\mathbb{E}_{\mathcal{P}}[f(\theta_{t_i})f(\theta_{t_j})] - \mathbb{E}_{\mathcal{P}'}[f(\theta_{t_i})f(\theta_{t_j})]| \leq \Delta, \\
|\mathbb{E}_{\mathcal{Q}}[f(\theta_{t_i})]\mathbb{E}[f(\theta_\infty)] - \mathbb{E}_{\mathcal{Q}'}[f(\theta_{t_i})]\mathbb{E}[f(\theta_\infty)]| \leq \Delta.
\end{aligned}$$

Then by applying triangle inequality twice:

$$|\mathbb{E}_{\mathcal{P}}[f(\theta_{t_i})f(\theta_{t_j})] - \mathbb{E}_{\mathcal{Q}}[f(\theta_{t_i})]\mathbb{E}[f(\theta_\infty)]| = O(\Delta)$$

Now we can prove (E3) as follows:

$$\begin{aligned}
&|\mathbf{Cov}\left(f(\theta_{t_i}), f(\theta_{t_j})\right)| \\
&= |\mathbb{E}[(f(\theta_{t_i}) - \mathbb{E}[f(\theta_{t_i})])(f(\theta_{t_j}) - \mathbb{E}[f(\theta_{t_j})])]| \\
&= |\mathbb{E}[f(\theta_{t_i})f(\theta_{t_j})] - \mathbb{E}[f(\theta_{t_i})]\mathbb{E}[f(\theta_{t_j})]| \\
&\leq |\mathbb{E}[f(\theta_{t_i})f(\theta_{t_j})] - \mathbb{E}[f(\theta_{t_i})]\mathbb{E}[f(\theta_\infty)]| + \\
&\qquad |\mathbb{E}[f(\theta_{t_i})]\mathbb{E}[f(\theta_\infty)] - \mathbb{E}[f(\theta_{t_i})]\mathbb{E}[f(\theta_{t_j})]| \\
&\leq O(\Delta) + |\mathbb{E}[f(\theta_\infty)] - \mathbb{E}[f(\theta_{t_j})]| = O(\Delta).
\end{aligned}$$

□

Proof of Theorem A.1. We again assume without loss of generality Δ is at most a sufficiently small constant. The proof strategy will be to express $\mathbb{E}[S]$ in terms of individual variances $\mathbf{Var}\left(f(\theta_{t_i})\right)$, which can be bounded using Lemma A.6.

We have the following:

$$\begin{aligned}
\mathbb{E}[S] &= \frac{\sum_{i=1}^k p_i^2}{k-1} \sum_{i=1}^k \mathbb{E}\left[(f(\theta_{t_i}) - \widehat{\mu})^2\right] \\
&= \frac{\sum_{i=1}^k p_i^2}{k-1} \sum_{i=1}^k \mathbb{E}\left[\left(\frac{k-1}{k}\right)^2 \left(\underbrace{f(\theta_{t_i})}_{x_i} - \underbrace{\frac{1}{k-1} \sum_{j \in [k], j \neq i} f(\theta_{t_j})}_{y_i}\right)^2\right].
\end{aligned} \tag{10}$$

From (10), we have the following:

$$\begin{aligned}
&\mathbb{E}\left[(x_i - y_i)^2\right] \\
&= \mathbb{E}[x_i^2] - 2\mathbb{E}[x_i y_i] + \mathbb{E}[y_i^2] \\
&= \left(\mathbb{E}[x_i^2] - (\mathbb{E}[x_i])^2\right) + \left(\mathbb{E}[y_i^2] - (\mathbb{E}[y_i])^2\right) + \\
&\qquad \left((\mathbb{E}[x_i])^2 + (\mathbb{E}[y_i])^2 - 2\mathbb{E}[x_i y_i]\right) \\
&= \underbrace{\mathbf{Var}(x_i)}_{A} + \underbrace{\mathbf{Var}(y_i)}_{B} + \underbrace{\left((\mathbb{E}[x_i])^2 + (\mathbb{E}[y_i])^2 - 2\mathbb{E}[x_i y_i]\right)}_{C}.
\end{aligned} \tag{11}$$

In the following, we bound each of the terms A, B, and C individually. First, let us consider the term B. We have the following:

$$\begin{aligned}
B = \mathbf{Var}(y_i) &= \frac{1}{(k-1)^2} \\
&\left(\sum_{j \in [k], j \neq i} \mathbf{Var}\left(f(\theta_{t_j})\right) + 2 \sum_{\substack{1 \leq j < \ell \leq k \\ j \neq i, \ell \neq i}} \mathbf{Cov}\left(f(\theta_{t_j}), f(\theta_{t_\ell})\right)\right).
\end{aligned} \tag{12}$$

Plugging Lemma A.6, (E3) into (12) we bound the variance of y_i as follows:

$$B = \mathbf{Var}(y_i) = \frac{1}{(k-1)^2} \left(\sum_{j \in [k], j \neq i} \mathbf{Var}\left(f(\theta_{t_j})\right)\right) \pm O(\Delta). \tag{13}$$

We now focus on bounding the term C in (11). Lemma A.6, (E1) and (E3) implies the following:

$$(\mathbb{E}[x_i])^2 = (\mathbb{E}[f(\theta_{t_k})])^2 \pm O(\Delta), \tag{14}$$

$$(\mathbb{E}[y_i])^2 = (\mathbb{E}[f(\theta_{t_k})])^2 \pm O(\Delta), \tag{15}$$

$$\mathbb{E}[x_i y_i] = (\mathbb{E}[f(\theta_{t_k})])^2 + O(\Delta). \tag{16}$$

Plugging (14),(15), and (16) into (11), we have

$$\mathbb{E}\left[(x_i - y_i)^2\right]$$

$$= \mathbf{Var}\left(f(\theta_{t_i})\right) + \frac{1}{(k-1)^2}\left(\sum_{j\in[k],j\neq i} \mathbf{Var}\left(f(\theta_{t_j})\right)\right) \pm O(\Delta). \tag{17}$$

Now, Lemma A.6, (E1) and (E2) implies

$$\forall i,: \left|\mathbf{Var}\left(f(\theta_{t_i})\right) - \frac{V}{\sum_{i=1}^k p_i^2}\right| = O(\Delta)$$

. So from (17) we have the following:

$$\mathbb{E}\left[(x_i - y_i)^2\right] = V \cdot \frac{k}{(k-1)\sum_{i=1}^k p_i^2} \pm O(\Delta). \tag{18}$$

Plugging this bound back in (10), we have the following:

$$\mathbb{E}[S] = \frac{\sum_{i=1}^k p_i^2}{k-1} \cdot \left(\frac{k-1}{k}\right)^2 \cdot k \cdot \left(V \cdot \frac{k}{(k-1)\sum_{i=1}^k p_i^2} \pm O(\Delta)\right)$$

$$= V \pm O(\Delta \sum_{i=1}^k p_i^2). \tag{19}$$

Which completes the proof. □

A.2 Optimizing the Number of Checkpoints

In Theorem A.1, we fixed the number of checkpoints and gave lower bounds on the burn-in time and separation between checkpoints needed for the sample variance bound to have bias at most Δ. We could instead consider the problem where T, the time of the final checkpoint, is fixed, and we want to choose k which minimizes the (upper bound on) mean squared error of the sample variance of $\{f(\theta_{iT/k})\}_{i\in[k]}$. Here, we sketch a solution to this problem using the bound from this section.

The mean squared error of the sample variance is the sum of the bias and variance of this estimator. We will use the following simplified reparameterization of Theorem A.1:

Theorem A.7 (Simpler version of Theorem A.1). *Let $c_1 := \frac{1}{2M} + \ln(c_2 M(p + \|\theta_0 - \theta^*\|_2^2))$, where c_2 is a sufficiently large constant. Then if S is the sample variance of $\{f(\theta_{iT/k})\}_{i\in[k]}$, V is the true variance of $f(\theta_T)$, and $T/k > c_1$:*

$$|\mathbb{E}[S] - V|^2 \leq \exp\left(-\frac{T/k - c_1}{c_2}\right).$$

One can also bound the variance of S:

Lemma A.8. *If $\bar{S}$ is the sample variance of $k > 1$ i.i.d. samples of θ_T, then if c_2 is a sufficiently large constant, for c_1 as defined in Lemma A.7:*

$$\mathbf{Var}\left(\bar{S}\right) \leq \frac{1}{k}, |\mathbf{Var}\left(S\right) - \mathbf{Var}\left(\bar{S}\right)| \leq 2\exp\left(-\frac{T/k - c_1}{c_2}\right).$$

Proof. Let $x_1, \ldots, x_k$ be k i.i.d. samples of $f(\theta_T)$, then since each x_i is in the interval $[-1, 1]$:

$$\mathbf{Var}\left(\bar{S}\right) = \frac{\mathbb{E}[x_1^4]}{k} - \frac{\mathbf{Var}\left(x_1\right)(k-3)}{k(k-1)} \leq \frac{1}{k}.$$

Giving the first part of the lemma. For the second part, let x_i be the sampled value of $f(\theta_{iT/k})$. Then:

$$\mathbb{E}[S^2] = \mathbb{E}\left[\left(\frac{1}{k-1}\sum_{i\in[k]}\left(x_i - \frac{1}{k}\sum_{j\in[k]} x_j\right)^2\right)^2\right].$$

For some coefficients $c_{i,j,\ell,m}$, this can be written as

$$\sum_{i\leq j\leq \ell\leq m} c_{i,j,\ell,m}\mathbb{E}[x_i x_j x_\ell x_m]$$

where $\sum_{i\leq j\leq \ell\leq m}|c_{i,j,\ell,m}| \leq 2$. By a similar argument to Theorem A.1, the change in this expectation if we instead use x_i that are i.i.d. is then at most $\exp\left(-\frac{T/k-c_1}{c_2}\right)$ as long as c_2 is a sufficiently large constant. In other words, $|\mathbb{E}[S^2] - \mathbb{E}[\bar{S}^2]| \leq \exp\left(-\frac{T/k-c_1}{c_2}\right)$. A similar argument applies to $E[S]^2$, giving the second part of the lemma. □

Putting it all together, we have an upper bound on the mean squared error of the sample variance of:

$$\frac{1}{k} + 3\exp\left(-\frac{T/k - c_1}{c_2}\right),$$

Assuming $k > 1, T/k > c_1$. Minimizing this expression with respect to k gives

$$k = \frac{T}{c_1 + c_2\ln(3T/c_2)},$$

which we can then round to the nearest integer larger than 1 to determine the number of checkpoints to use that minimizes our upper bound on the mean squared error. Of course, if $T < 2c_1$ then Theorem A.1 cannot be applied to give a meaningful bias bound for any number of checkpoints, so this choice of k is not meaningful in that case.

A.3 Proof of Lemma A.4

We will bound the divergences $D_\alpha(P_1||P_2), D_\alpha(P_2||P_3), D_\alpha(P_3||P_4)$ where P_1 is the distribution θ_η that is the solution to (9), P_2 is a Gaussian centered at the point $\theta_0 - \eta\nabla\mathcal{L}(\theta_0; D)$, P_3 is a Gaussian centered at θ^*, and P_4 is the stationary distribution of (9). Then, we can use the approximate triangle inequality for Rényi divergences to convert these pairwise bounds into the desired bound.

Lemma A.9. *Fix some θ_0. Let P_1 be the distribution of θ_η that is the solution to (9), and let P_2 be the distribution $N(\theta_0 - \eta\nabla\mathcal{L}(\theta_0; D), 2\eta)$. Then:*

$$D_\alpha(P_1||P_2) = O\left(M^2\ln(\alpha)\cdot\max\{p\eta^2, \|\theta_0 - \theta^*\|_2^2\eta^3\}\right)$$

Proof. Let θ_t be the solution trajectory of (9) starting from θ_0, and let θ_t' be the solution trajectory if we replace $\nabla\mathcal{L}(\theta_t; D)$ with $\nabla\mathcal{L}(\theta_0; D)$. Then θ_η is distributed according to P_1 and θ_η' is distributed according to P_2.

By a tail bound on Brownian motion (see e.g. Fact 32 in [42]), we have that $\max_{t\in[0,\eta]} \left\|\int_0^t dW_s ds\right\|_2 \leq \sqrt{\eta(p + 2\ln(2/\delta))}$ w.p. $1-\delta$. Then following the proof of Lemma 13 in [42], w.p. $1-\delta$,

$$\max_{t\in[0,\eta]} \|\theta_t - \theta_0\|_2 \leq cM(\sqrt{p} + \sqrt{\ln(1/\delta)})\sqrt{\eta} + M\|\theta_0 - \theta^*\|_2\,\eta,$$

for some sufficiently large constant c, and the same is true w.p. $1-\delta$ over θ'_t. Now, following the proof of Theorem 15 in [42], for some constant c', we have the divergence bound $D_\alpha(P_1||P_2) \leq \varepsilon$ as long as:

$$\frac{M^4 \ln^2\alpha}{\varepsilon^2}(p\eta^2 + \|\theta_0 - \theta^*\|_2^2\,\eta^3) < c'.$$

In other words, for any fixed η, we get a divergence bound of:

$$D_\alpha(P_1||P_2) = O\left(M^2\ln(\alpha)\cdot\max\{p\eta^2, \|\theta_0-\theta^*\|_2^2\,\eta^3\}\right),$$

as desired. □

Lemma A.10. *Let P_2 be the distribution $N(\theta_0 - \eta\nabla\mathcal{L}(\theta_0;D), 2\eta)$ and P_3 be the distribution $N(\theta^*, 2\eta)$. Then for $\eta \leq 2/M$:*

$$D_\alpha(P_2||P_3) \leq \frac{\alpha\|\theta_0-\theta^*\|_2^2}{4\eta}.$$

Proof. By contractivity of gradient descent we have:

$$\|\theta_0 - \eta\nabla\mathcal{L}(\theta_0;D) - \theta^*\|_2 \leq \|\theta - \theta^*\|_2.$$

Now the lemma follows from Rényi divergence bounds between Gaussians (see e.g., Example 3 of [84]). □

Lemma A.11. *Let P_3 be the distribution $N(\theta^*, 2\eta)$ and let P_4 be the stationary distribution of (9). Then for $\eta \leq 1/2M$ we have:*

$$D_\alpha(P_3||P_4) \leq \frac{\alpha}{\alpha-1}\left(\frac{p}{2}\ln(1/\eta) - \ln(2\pi)\right) + \frac{p}{2}\ln(\alpha/4\pi\eta).$$

Proof. We have $P_3(\theta) = P_3(\theta^*)\exp(-\frac{1}{4\eta}\|\theta-\theta^*\|_2^2)$ where $P_3(\theta^*) = \left(\frac{1}{4\pi\eta}\right)^d$. By M-smoothness of the negative log density of P_4, we also have $P_4(\theta) \geq P_4(\theta^*)\exp(-\frac{M}{2}\|\theta-\theta^*\|_2^2)$. In addition, since P_4 is 1-strongly log concave, $P_4(\theta^*) \geq \left(\frac{1}{2\pi}\right)^{p/2}$ (as the 1-strongly log concave density with mode θ^* that minimizes $P_4(\theta^*)$ is the multivariate normal with mean θ^* and identity covariance). Finally, for $\alpha \geq 1$ and $\eta \leq 1/2M$, we have $\alpha/4\eta > (\alpha-1)M/2$. Putting it all together:

$$\exp((\alpha-1)D_\alpha(P_3||P_4)) \tag{20}$$

$$= \int \frac{P_3(\theta)^\alpha}{P_4(\theta)^{\alpha-1}}d\theta \tag{21}$$

$$= \frac{P_3(\theta^*)^\alpha}{P_4(\theta^*)^{\alpha-1}}\times \int \exp\left(-(\frac{\alpha}{4\eta} - (\alpha-1)\frac{M}{2})\|\theta-\theta^*\|_2^2\right)d\theta \tag{22}$$

$$\leq \left(\frac{1}{4\pi\eta}\right)^{\alpha p/2}(2\pi)^{\alpha(p-1)/2}\times \int \exp\left(-(\frac{\alpha}{4\eta} - (\alpha-1)\frac{M}{2})\|\theta-\theta^*\|_2^2\right)d\theta \tag{23}$$

$$= \left(\frac{1}{2\pi}\right)^{\alpha/2}\left(\frac{1}{2\eta}\right)^{\alpha p/2}\times \int \exp\left(-(\frac{\alpha}{4\eta} - (\alpha-1)\frac{M}{2})\|\theta-\theta^*\|_2^2\right)d\theta \tag{24}$$

$$\overset{(*)}{=} \left(\frac{1}{2\pi}\right)^{\alpha/2}\left(\frac{1}{2\eta}\right)^{\alpha p/2}\left(\frac{\frac{\alpha}{4\eta} - (\alpha-1)\frac{M}{2}}{\pi}\right)^{p/2} \tag{25}$$

$$\leq \left(\frac{1}{2\pi}\right)^{\alpha/2}\left(\frac{1}{2\eta}\right)^{\alpha p/2}\left(\frac{\alpha}{4\pi\eta}\right)^{p/2} \tag{26}$$

$$\implies D_\alpha(P_3||P_4) \leq \frac{\alpha}{\alpha-1}\left(\frac{p}{2}\ln(1/\eta) - \ln(2\pi)\right) + \frac{p}{2}\ln(\alpha/4\pi\eta). \tag{27}$$

In $(*)$, we use the fact that $\alpha/4\eta > (\alpha-1)M/2$ to ensure the integral converges. □

Lemma A.12. *Fix some point θ_0. Let P be the distribution θ_η that is the solution to (9) from θ_0 for time $\eta \leq 1/2M$. Let Q be the stationary distribution of (9). Then:*

$$D_\alpha(\mathcal{P}||\mathcal{Q}) = O\left(M^2\ln(\alpha)\cdot\max\{p\eta^2, \|\theta_0-\theta^*\|_2^2\,\eta^3\} + \frac{\alpha\|\theta_0-\theta^*\|_2^2}{\eta} + p\ln(\alpha/\eta).\right)$$

Proof. By monotonicity of Rényi divergences (see e.g., Proposition 9 of [58]), we can assume $\alpha \geq 2$. Then by applying twice the approximate triangle inequality for Rényi divergences (see e.g. Proposition 11 of [58]), we get:

$$D_\alpha(P_1||P_4) \leq \frac{5}{3}D_{3\alpha}(P_1||P_2) + \frac{4}{3}D_{3\alpha-1}(P_2||P_3) + D_{3\alpha-2}(P_3||P_4).$$

The lemma now follows by Lemmas A.9, A.10, A.11. □

Lemma A.4 now follows by plugging $\alpha = 2, \eta = 1/2M$ into Lemma A.12 and then using Theorem 2 of [85].

A.4 Extending to DP-SGLD

While we presented our results in terms of DP-LD to simplify the results, a similar result can be proven for DP-SGLD, which is a discrete algorithm and just a reparameterization of DP-SGD, the algorithm we use in our experiments. So, our results can still be

Table 6: StackOverflow LSTM architecture details.

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

applied to some practical settings. We discuss how to modify the proof of Theorem A.1 here.

The only part of the proof of Theorem A.1 which does not immediately hold (or hold in an analogous form) for DP-SGLD is Lemma A.4. That is, if we can show that starting from a point distribution, we converge to the stationary distribution of DP-LD in a given number of iterations of DP-SGLD, then we can prove an analog of Lemma A.4 and the rest of the proof of Theorem A.1 can be used as-is.

To prove an analog of Lemma A.4, we need (i) an analog of Lemma A.12, which shows that from a point distribution we reach a finite Renyi divergence from the stationary distribution and (ii) an analog of Theorem 2 of [85], which shows that from a finite Renyi divergence bound we can reach a small Renyi divergence bound in a given amount of time.

(i) Can be proven similarly to Lemma A.12; in particular, we only need Lemmas A.10 and A.11, which by triangle inequality give a Renyi divergence bound between the distribution given after one iteration of DP-SGLD from a point distribution and the stationary distribution. (ii) can be proven using e.g. Lemma 7 of [32], which shows how the Renyi divergence decreases in every iteration under the assumptions in this section. Getting an exact lower bound on the number of iterations of DP-SGLD needed analogous to our lower bounds on κ_1, κ_2 requires a bit of technical work and results in a much more complicated bound than Theorem A.1, so we omit the details here. However, we note that an analogous version of one of our high-level takeaways from Theorem A.1, that κ_1 can be much larger than κ_2 in the worst case, would hold for the bounds we could prove for DP-SGLD. In particular, it is still the case that the initial divergence we get from (i) depends on the distance to the minimizer of $\mathcal{L}$, which can be arbitrarily bad for the initialization but which we can bound with high probability for the intermediate checkpoints via Lemma A.4.

B Missing details from Section 4

Below we provide some preliminaries, details about the experimental setup, and results that were omitted from Section 4 due to space constraints.

Table 7: Tuning k, the number of past checkpoints used for aggregation in $\mathrm{UTA}_{\mathrm{inf}}$, OPA, OMV. Results below are for original CIFAR10 dataset.

Aggregation	Privacy level	Number of past checkpoints aggregated, k											
		3	5	10	20	30	40	50	60	70	80	90	100
$\mathrm{UTA}_{\mathrm{inf}}$	$\varepsilon = 8$	78.07	78.24	78.37	78.81	**79.39**	79.16	79.31	79.13	79.01	78.88	78.79	78.71
	$\varepsilon = 1$	55.29	55.72	55.96	56.12	56.43	**56.62**	56.59	56.48	56.33	56.38	56.4	56.26
OPA	$\varepsilon = 8$	78.05	78.22	78.45	78.89	79.18	79.33	**79.40**	79.40	79.36	79.37	79.26	79.11
	$\varepsilon = 1$	55.75	56.11	56.23	56.49	**56.68**	56.11	56.43	56.11	55.91	55.94	55.80	55.81
OMV	$\varepsilon = 8$	76.77	77.2	77.7	78.32	78.65	78.81	78.70	78.92	79.21	79.20	**79.34**	79.31
	$\varepsilon = 1$	53.75	54.51	54.90	55.10	55.07	55.78	56.14	56.33	**56.40**	56.19	56.23	56.00

Table 8: Training hyperparameters that we use for StackOverflow experiments with DP-FTRL [28] and various training (Section 3.1) and inference (Section 3.2) aggregations. We use the hyperparameters in "Baseline" section for all the inference aggregations; we discuss how we tune individual parameters of the aggregations in Section 3.3

Aggregation	Privacy	Parameter	clip norm	noise multiplier	server lr	client lr	server momentum
Baseline	$\varepsilon = \infty$	–	1.0	0.0	3.0	0.5	0.9
	$\varepsilon = 18.9$	–	1.0	0.341	0.5	1.0	0.95
	$\varepsilon = 8.2$	–	1.0	0.682	0.25	1.0	0.95
$\mathrm{UPA}_{\mathrm{tr}}$	$\varepsilon = \infty$	$k = 3$	1.0	0.0	2.0	0.5	0.95
	$\varepsilon = 18.9$	$k = 3$	0.3	0.341	2.0	1.0	0.95
	$\varepsilon = 8.2$	$k = 3$	0.3	0.682	1.0	1.0	0.95
$\mathrm{EMA}_{\mathrm{tr}}$	$\varepsilon = \infty$	$\beta = 0.95$	1.0	0.0	2.0	1.0	0.95
	$\varepsilon = 18.9$	$\beta = 0.95$	1.0	0.341	0.5	1.0	0.95
	$\varepsilon = 8.2$	$\beta = 0.95$	1.0	0.682	0.25	1.0	0.95

Table 9: Training hyperparameters that we use for *periodic distribution shifting StackOverflow* experiments with DP-FTRL [28] and various training (Section 3.1) and inference (Section 3.2) aggregations. We use the hyperparameters in "Baseline" section for all the inference aggregations; we discuss how we tune individual parameters of the aggregations in Section 3.3

Aggregation	Privacy ε	Parameter	clip norm	noise multiplier	server lr	client lr	server momentum
Baseline	∞	–	1.0	0.0	3.0	0.5	0.9
	18.9	–	1.0	0.341	0.5	1.0	0.95
	8.2	–	1.0	0.682	0.25	1.0	0.95
$\mathrm{UPA}_{\mathrm{tr}}$	∞	$k = 5$	1.0	0.0	2.0	0.5	0.95
	18.9	$k = 5$	1.0	0.341	0.5	1.0	0.95
	8.2	$k = 5$	0.3	0.682	1.0	0.5	0.95
$\mathrm{EMA}_{\mathrm{tr}}$	∞	$\beta = 0.95$	1.0	0.0	2.0	0.5	0.95
	18.9	$\beta = 0.95$	1.0	0.341	0.5	1.0	0.95
	8.2	$\beta = 0.95$	1.0	0.682	1.0	0.5	0.95

Table 10: Training hyperparameters that we use for CIFAR10 and periodic distribution shifting (PDS) CIFAR10 experiments with DP-SGD [28] and various training aggregations (Section 3.1); we discuss how we tune individual parameters of the aggregations in Section 3.3

Aggregation	Privacy	Parameter	noise multiplier	learning rate	τ (T)
CIFAR10; DP-SGD; sample-level privacy					
UTA_{tr}	$\varepsilon = 8$	$k = 2$	3.0	4.0	2000 (3068)
	$\varepsilon = 1$	$k = 2$	8.0	2.0	400 (568)
EMA_{tr}	$\varepsilon = 8$	$\beta = 0.6$	4.0	2.0	2000 (4559)
	$\varepsilon = 1$	$\beta = 0.5$	10.0	2.0	400 (875)
PDS CIFAR10; DP-SGD; sample-level privacy					
UTA_{tr}	$\varepsilon = 8$	$k = 5$	3.0	2.0	2000 (2480)
	$\varepsilon = 1$	$k = 3$	8.0	2.0	400 (460)
EMA_{tr}	$\varepsilon = 8$	$\beta = 0.6$	3.0	2.0	1500 (2480)
	$\varepsilon = 1$	$\beta = 0.6$	8.0	2.0	200 (460)

Table 11: Training hyperparameters that we use for CIFAR100 and periodic distribution shifting (PDS) CIFAR100 experiments with DP-SGD [28] and various training aggregations (Section 3.1); we discuss how we tune individual parameters of the aggregations in Section 3.3

Aggregation	Privacy	Parameter	noise multiplier	learning rate	τ (T)
CIFAR100; DP-SGD; sample-level privacy					
UTA_{tr}	$\varepsilon = 8$	$k = 50$	9.4	4.0	400 (2000)
	$\varepsilon = 1$	$k = 50$	21.1	4.0	100 (250)
EMA_{tr}	$\varepsilon = 8$	$\beta = 0.85$	9.4	4.0	1500 (2000)
	$\varepsilon = 1$	$\beta = 0.99$	21.1	4.0	200 (250)
PDS CIFAR100; DP-SGD; sample-level privacy					
UTA_{tr}	$\varepsilon = 8$	$k = 10$	9.4	4.0	50 (2000)
	$\varepsilon = 1$	$k = 5$	21.1	4.0	200 (250)
EMA_{tr}	$\varepsilon = 8$	$\beta = 0.85$	9.4	4.0	200 (2000)
	$\varepsilon = 1$	$\beta = 0.85$	21.1	4.0	200 (250)