Differentially Private Model Publishing for Deep Learning

Lei Yu, Ling Liu, Calton Pu, Mehmet Emre Gursoy, Stacey Truex
School of Computer Science, College of Computing, Georgia Institute of Technology
Email: leiyu@gatech.edu, {ling.liu,calton.pu}@cc.gatech.edu, {memregursoy,staceytruex}@gatech.edu

Abstract—Deep learning techniques based on neural networks have shown significant success in a wide range of AI tasks. Large-scale training datasets are one of the critical factors for their success. However, when the training datasets are crowdsourced from individuals and contain sensitive information, the model parameters may encode private information and bear the risks of privacy leakage. The recent growing trend of the sharing and publishing of pre-trained models further aggravates such privacy risks. To tackle this problem, we propose a differentially private approach for training neural networks. Our approach includes several new techniques for optimizing both privacy loss and model accuracy. We employ a generalization of differential privacy called concentrated differential privacy(CDP), with both a formal and refined privacy loss analysis on two different data batching methods. We implement a dynamic privacy budget allocator over the course of training to improve model accuracy. Extensive experiments demonstrate that our approach effectively improves privacy loss accounting, training efficiency and model quality under a given privacy budget.

Keywords-differential privacy; stochastic gradient descent; deep learning; model publishing; privacy budget allocation

I. Introduction

In recent years, deep learning techniques based on artificial neural networks have dramatically advanced the state of the art in a wide range of AI tasks such as speech recognition, image classification, natural language processing and game playing. Its success relies on three sources of advancement: high-performance computing, large-scale datasets, and the increasing number of open source deep learning frameworks, such as TensorFlow, Caffe, and Torch.

Privacy Concerns in Deep Learning. However, recent studies on membership attacks and model inversion attacks have exposed potential privacy risks from a number of dimensions. First, large-scale datasets are collected from individuals via crowdsourcing platforms, containing private information such as location, images, medical, and financial data of the users. The users usually do not have any control over how their data is being used or shared once collected. Second, deep neural networks have a large number of hidden layers, leading to a large effective capacity that could be sufficient for encoding the details of some individual's data into model parameters or even memorizing the entire dataset [41]. It has been shown that individual information can be effectively extracted from neural networks [19], [34]. Therefore, there are severe privacy concerns accompanied

with the broad deployment of deep learning applications and deep learning as a service .

On the other hand, the publishing and sharing of trained deep learning models has been gaining growing interest. Google's cloud machine learning services provide several pre-trained models usable out-of-the-box through a set of APIs. The model owners can also publish their trained models to the cloud and allow other users to perform predictions through APIs. In mobile applications, entire models are stored on-device to enable power-efficient and low-latency inference. Transfer learning [40], a key technique of deep learning, can leverage and adapt the already existing models to new classes of data, saving the effort of training the entire neural network from scratch. People who only have small datasets can use the model trained on a large dataset as a fixed feature extractor in their neural networks or adapt the model to their own domain. Transfer learning is believed to be the next driver of machine learning success in industry and will significantly stimulate the sharing of pre-trained models. A large number of pre-trained models have been publicly available in model zoo repositories [6]. In these cases, the model parameters are entirely exposed, making it easier for adversaries to launch inference attacks, such as membership attacks [34] or model inversion attacks [19], to infer sensitive data records of individuals in the training datasets. Even by providing only the query APIs to access remote trained models, the model parameters may still be extracted from prediction queries and in turn used to infer the sensitive training data [38]. Therefore, it is imperative to develop principled privacy-preserving deep learning techniques to protect private training data against adversaries with full knowledge of model parameters.

Deep learning with Differential Privacy. Although privacy-preserving machine learning has attracted much attention over the last decade, privacy preserving deep learning was first proposed in 2015 [33]. The proposed approach argues for privacy-preserving model training in a collaborative federated learning system and involves multiple participants jointly training a model by sharing sanitized parameters while keeping their training data private and local. The first proposal for deep learning with differential privacy was presented in 2016 [7]. Differential privacy (DP), a defacto standard for privacy that offers provable privacy guarantees, has been applied for privacy-preserving machine learning [8], [10], [11], [32], [42]. DP characterizes the

difference in output between two input datasets differing by at most one element. This characterization is challenging with deep learning because the internal representations of deep neural networks are notoriously difficult to understand. Prior works [7], [28], [37] suggest using the norm gradient clipping in the stochastic gradient descent (SGD) algorithm to bound the influence of any single example on the gradients and then applying differentially private mechanisms to perturb the gradients accordingly. By ensuring that each gradient descent step is differentially private, the final output model satisfies a certain level of differential privacy given the composition property. It is known that the SGD training process of a deep neural network tends to involve a large number of iterations. Given a target differential privacy guarantee, the differentially private training algorithm needs a tight estimation on the privacy loss for the composition of DP. This is necessary for the algorithm to effectively track cumulative privacy loss during the training process and, if necessary, terminate before the loss exceeds the privacy budget. Unfortunately, Abadi and his co-authors [7] have shown the existing strong composition theorem [18] for differential privacy does not yield a tight analysis. To address this problem, the moments accountant method is proposed [7], which tracks the log moments of the privacy loss variable and provides a much tighter estimate of the privacy loss for the composition of Gaussian mechanisms under random sampling.

In this paper, however, we analyze several issues with using the differentially private SGD (DP-SGD) algorithm and privacy accounting method proposed in [7]. The first problem is related to the underestimation of privacy loss caused by data batching methods. For computational efficiency, the SGD algorithm usually takes small batches from the training dataset each to iteratively compute gradients and update model parameters. The DP-SGD approach in [7] exploits the privacy amplification of random sampling to produce a tighter estimation of privacy loss. This is based on the assumption that the data batches for mini-batch SGD input are generated through random sampling with replacement on the training dataset. In practice, for better efficiency, the data batching method is implemented through random reshuffling, which randomly shuffles the training dataset and then partitions them into batches of the similar size [5], [22]. We note that the random sampling and the random reshuffling are two different implementation methods for data batching, and our analysis and experiments show that they cause distinct privacy loss. Therefore, the composition of differentially private mechanisms depends on how data is accessed by each mechanism. Simply treating reshuffling and random sampling as the same data access will lead to the underestimation of privacy loss.

The second issue is the need for a tight analysis on cumulative privacy loss for DP-SGD which tends to have a large number of iterations. To address this problem, we propose the use of concentrated differential privacy (CDP), a generalization of differential privacy recently introduced by Dwork and Rothblum [17]. CDP focuses on cumulative privacy loss for a large number of computations and provides a sharper analysis tool. Based on CDP, we analyze the privacy loss under different data batching methods and develop privacy accounting methods for each respectively. In particular, our analysis for random reshuffling provides a tighter estimation on privacy loss than the strong composition theorem even when exploiting the privacy amplification effect of random sampling. For the random sampling based batching method, we show that CDP is not able to capture the privacy amplification effect of random sampling. We address this problem by using a relaxation and conversion to traditional (ϵ, δ)-differential privacy. Compared with the moments accountant method which requires numerical computation of log moments for a range of moment orders, our privacy accounting method produces a very close estimation under random sampling with simpler and faster calculation.

The third novelty of our approach to differentially private deep learning is our development of a dynamic privacy budget allocation to improve model accuracy under differentially private training. The perturbed gradients during the training process inevitably degrade model accuracy. We aim to provide differential privacy guarantees on the final output models. This means that we are much less concerned with the privacy loss of a single iteration. This provides opportunities to optimize the model accuracy via adjusting privacy budget allocation for every training iteration. In this paper, we propose a set of dynamic privacy budget allocation methods and our extensive experiments demonstrate benefits for improving the model accuracy. It is worth noting that the techniques proposed in this paper not only apply to neural networks but also may apply to any other iterative learning algorithms.

The remainder of the paper is as follows. We review necessary background in Section II, provide an overview of our approach in Section III, and then give a detailed technical development in Section IV. Section V describes experimental results. Some discussions are given in Section VI. Section VII presents related work and Section VIII concludes the paper. Deferred proofs are provided in the Appendix.

II. BACKGROUND

A. Deep Learning

Deep learning uses neural networks that are defined as a hierarchical composition of parameterized functions to model the input data. For supervised learning, the training data are labeled with correct classes, and a multi-layer neural network is deployed to model the correlation between data instances and their labels. A typical neural network consists of $n(n > 1)$ layers of neurons. Each layer of neurons is parameterized by a weight matrix $W^{(l)}$ and a bias vector

$b^{(l)}$. Layers apply an affine transformation to the previous layer's output and then computes an activation function σ over that. Typical examples of the activation function σ are sigmoid, rectified linear unit(ReLU) and tanh.

The training of a neural network aims to learn the parameters $\theta = \{W^{(l)}, b^{(l)} | 1 \leq l \leq n\}$ that minimize a loss function L defined to represent the penalty for misclassifying the training data. It is usually a non-convex optimization problem and solved by gradient descent. The gradient descent method iteratively computes the gradient of the loss function L and updates the parameters every step until the loss converges to a local optimum. In practice, the training of neural networks uses the mini-batch stochastic gradient descent (SGD) algorithm, which is much more efficient for large datasets. At each step a batch B of examples is sampled from the training dataset and the gradient of the average loss is computed, i.e., $\frac{1}{|B|} \sum_{x \in B} \nabla_\theta L(\theta, x)$ as $\nabla_\theta L(\theta)$. The SGD algorithm then applies the following update rule for parameters θ

$$\theta = \theta - \alpha \nabla_\theta L(\theta) \qquad (1)$$

where α is the learning rate. The running time of the mini-batch SGD algorithm is usually expressed as the number of *epochs*. Each epoch consists of all of the batches of the training dataset, i.e., in an epoch every example has been seen once. Within an epoch, the pass of one batch of examples for updating the model parameters is called one *iteration*.

B. Differential Privacy

Differential privacy is a rigorous mathematical framework that formally defines the privacy properties of data analysis algorithms. Informally it requires that any changes to a single data point in the training dataset can only cause statistically insignificant changes to the algorithm's output.

Definition 1 (Differential Privacy [14]). *A randomized mechanism $\mathcal{A}$ provides (ϵ, δ)-differential privacy if for any two neighboring database D and D' that differ in only a single entry, $\forall S \subseteq Range(\mathcal{A})$,*

$$\Pr(\mathcal{A}(D) \in S) \leq e^\epsilon \Pr(\mathcal{A})(D') \in S) + \delta \qquad (2)$$

If $\delta = 0$, $\mathcal{A}$ is said to be ϵ-differential privacy. In the rest of this paper, we write (ϵ, δ)-DP for short.

The standard approach to achieving differential privacy is the sensitivity method [14], [15] that adds to the output some noise that is proportional to the sensitivity of the query function. The sensitivity measures the maximum change of the output due to the change of a single database entry.

Definition 2 (Sensitivity [15]). *The sensitivity of a query function $q : \mathcal{D} \to \mathbb{R}^d$ is*

$$\Delta = \max_{D, D'} ||q(D) - q(D')|| \qquad (3)$$

where D, $D' \in \mathcal{D}$ are any two neighboring datasets that

differ at most one element, $|| \cdot ||$ denotes L_1 or L_2 norm.

In this paper, we choose the Gaussian mechanism that uses L_2 norm sensitivity. It adds zero-mean Gaussian noise with variance $\Delta^2 \sigma^2$ in each coordinate of the output $q(D)$, as

$$q(D) + \mathcal{N}(0, \Delta^2 \sigma^2 \mathbf{I}) \qquad (4)$$

It satisfies (ϵ, δ)-DP if $\sigma^2 > 2\log(\frac{1.25}{\delta})/\epsilon^2$ and $\epsilon \in (0, 1)$ [16].

C. Concentrated Differential Privacy

Concentrated differential privacy (CDP) is a generalization of differential privacy recently introduced by Dwork and Rothblum [17]. It aims to make privacy-preserving algorithms more practical for large numbers of computations than traditional DP while still providing strong privacy guarantees. It allows the computation to have much less concern about single-query loss but high probability bounds for the cumulative loss, and provides sharper and more accurate analysis on the cumulative loss for multiple computations compared to the popular (ϵ, δ)-DP.

CDP considers privacy loss on an outcome o as a random variable when the randomized mechanism $\mathcal{A}$ operates on two adjacent database D and D':

$$L^{(o)}_{(\mathcal{A}(D) || \mathcal{A}(D'))} \triangleq \log \frac{\Pr(\mathcal{A}(D) = o)}{\Pr(\mathcal{A}(D') = o)} \qquad (5)$$

The (ϵ, δ)-DP guarantee ensures that the privacy loss variable is bounded by ϵ but exceeds that with probability no more than δ. As a relaxation to that, (μ, τ)-concentrated differential privacy, (μ, τ)-CDP for short [17], ensures that the mean (i.e., expectation) of the privacy loss is no more than μ and the probability of the loss exceeding its mean by an amount of $t \cdot \tau$ is bounded by $e^{-t^2/2}$. An alternative formulation of CDP to Dwork and Rothblum's (μ, τ)-CDP is proposed by Bun and Steinke [9], called "zero-concentrated differential privacy" (zCDP for short). Instead of mean concentrated as (μ, τ)-CDP, zCDP makes privacy loss concentrated around zero (hence the name), still following sub-Gaussian such that larger deviations from zero become increasingly unlikely.

Definition 3 (Zero-Concentrated Differential Privacy (zCDP) [9]). *A randomized mechanism $\mathcal{A}$ is ρ-zero concentrated differentially private (i.e., ρ-zCDP) if for any two neighboring databases D and D' that differ in only a single entry and all $\alpha \in (1, \infty)$,*

$$D_\alpha(\mathcal{A}(D) || \mathcal{A}(D')) \triangleq \frac{1}{\alpha - 1} \log \left(\mathbb{E} \left[e^{(\alpha-1)L^{(o)}} \right] \right) \leq \rho\alpha \qquad (6)$$

Where $D_\alpha(\mathcal{A}(D) || \mathcal{A}(D'))$ called α-Rényi divergence between the distributions of $\mathcal{A}(D)$ and $\mathcal{A}(D')$.

The (ϵ, δ)-DP bounds the privacy loss by ensuring $\Pr(L^{(o)} > \epsilon) \leq \delta$. In contrast, zCDP entails a bound on the

moment generating function of privacy loss $L^{(o)}$, indicated by an equivalent form of (6)

$$\mathbb{E}\left[e^{(\alpha-1)L^{(o)}}\right] \le e^{(\alpha-1)\alpha\rho} \qquad (7)$$

This implies that for zCDP, privacy loss $L^{(o)}$ is assumed to be a sub-Gaussian random variable such that it has a strong tail decay property, namely, $\Pr(L^{(o)} > t+\rho) \le e^{-t^2/(4\rho)}$ for all $t > 0$ [9]. The following propositions are restatements of some zCDP results given in [9] that will be used in our paper.

Proposition 1. *If $\mathcal{A}$ provides ρ-zCDP, then $\mathcal{A}$ provides $(\rho + 2\sqrt{\rho\log(1/\delta)}, \delta)$-DP for any $\delta > 0$.*

Proposition 2. *The Gaussian mechanism with noise $\mathcal{N}(0, \Delta^2\sigma^2\mathbf{I})$ satisfies $(\frac{1}{2\sigma^2})$-zCDP.*

We use zCDP instead of original (μ, τ)-CDP because zCDP is comparable to (ϵ, δ)-DP, as indicated by Proposition 1 and is immune to post-processing while (μ, τ)-CDP is not closed under post-processing [9].

D. Composition

Differential privacy offers elegant composition properties that enable more complex algorithms and data analysis task via the composition of multiple differentially private building blocks. The composition should have privacy guarantees degraded gracefully with multiple outputs that may be subjected to the joint analysis from building blocks.

For a sequential composition of k mechanisms $\mathcal{A}_1, \ldots, \mathcal{A}_k$ satisfying (ϵ_i, δ_i)-DP for $i=1, \ldots, k$ respectively, the basic composition result [16] shows that the privacy composes linearly, i.e., the sequential composition satisfies $(\sum_i^k \epsilon_i, \sum_i^k \delta_i)$-DP. When $\epsilon_i = \epsilon$ and $\delta_i = \delta$, the strong composition bound from [18] states that the composition satisfies $(\epsilon\sqrt{2k\log(1/\delta')} + k\epsilon(e^\epsilon - 1), k\delta + \delta')$-DP. For zCDP, it has a simple linear composition property [9]:

Theorem 1. *Two randomized mechanisms $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy ρ_1-zCDP and ρ_2-zCDP respectively, their sequential composition $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ satisfies $(\rho_1 + \rho_2)$-zCDP.*

Compared with (ϵ, δ)-DP, CDP provides a tighter bound on the cumulative privacy loss under composition, which makes it more suitable for algorithms running a large number of iterations. In other words, while providing the same privacy guarantee, CDP allows lower noise scale and thus better accuracy. Consider k iterative composition of a Gaussian mechanism with noise $\mathcal{N}(0, \sigma^2\mathbf{I})$. To guarantee the final (ϵ, δ)-DP, in terms of (ϵ_i, δ_i)-DP for every iteration, the permitted loss ϵ_i of each iteration is $\epsilon_i = \epsilon/(2\sqrt{2k\log(1/(\delta - k\delta_i))})$, which will be very low when k is large. The noise scale is $\sigma = \frac{\sqrt{k}}{\epsilon}(4\sqrt{\log(1.25/\delta_i)\log(1/(\delta - k\delta_i))})$. Suppose $\delta_i = \delta/(k+1)$, we then have $\sigma > \frac{\sqrt{k}}{\epsilon}(4\log((k+1)/\delta))$. In contrast, with using ρ_i-zCDP for every iteration, because

$\rho \approx \epsilon^2/(4\log(1/\delta))$ zCDP satisfies (ϵ, δ)-DP [9] and $\rho_i = \frac{1}{k}\rho$. It is easy to show that the noise scale $\sigma = \frac{\sqrt{k}}{\epsilon}\sqrt{2log(1/\delta)}$ which is multiple times smaller than the noise scale derived under (ϵ, δ)-DP. On the other hand, a single parameter ρ of zCDP and its linear composition naturally fit the concept of a privacy budget. Thus, zCDP is an appropriate choice for privacy accounting.

III. OVERVIEW

Because it is difficult to characterize the maximum difference of the model parameters over any two neighboring datasets for neural networks, differentially private deep learning [7], [28], [37] relies on differentially private stochastic gradient descent (DP-SGD) to control the influence of training data on the model. This approach explicitly bounds per-example gradients $\nabla_\theta L(\theta, x)$ in every iteration by clipping the L_2 norm of gradient vectors. Given a clipping threshold C, this is done by replacing the gradient vector $\mathbf{g}$ with $\mathbf{g}/\max(1, \frac{\|\mathbf{g}\|L_2}{C})$ which scales $\mathbf{g}$ down to norm C if $\|\mathbf{g}\|_2 > C$. A Gaussian mechanism with L_2 norm sensitivity of C is then applied to perturb the gradients before the gradient descent step in Eq. (1) updates the model parameters. Because each SGD step is differentially private, by the composition property of differential privacy, the final model parameters are also differentially private. The problem with DP-SGD is that the training of a deep neural network (DNN) tends to have a large number of iterations, which causes large cumulative privacy loss at the end. Therefore, a tight estimation of privacy loss under composition is critical for allowing lower noise scale or more training iterations (for desired accuracy) when we have a fixed privacy budget.

To analyze the cumulative privacy loss of DP-SGD, we employ concentrated differential privacy(CDP) which was developed to accommodate a larger number of computations and provides sharper and tighter analysis of privacy loss than the strong composition theorem of (ϵ, δ)-DP. One way to track the privacy loss of DP-SGD is the Moments Accountant (MA) method proposed by Abadi et al. [7]. It assumes that the data batches for mini-batch SGD are generated by randomly sampling examples from the training dataset with replacement, MA takes advantage of the privacy amplification effect of random sampling to achieve a much tighter estimate on privacy loss than the strong composition theorem. It has been shown in [27] that running an (ϵ, δ) differentially private mechanism over a set of examples each of which is independently sampled with probability q $(0 < q < 1)$ achieves $(\log(1 + q(e^\epsilon - 1)), q\delta)$-DP. However, in practice, random batches are generated by randomly shuffling examples and then partitioning them into batches for computation efficiency, which is distinct from random sampling with replacement. By analyzing the privacy loss under these two data batching methods, random sampling with replacement and random reshuffling respectively, we show that 1) random sampling with replacement and random

reshuffling result in different privacy loss; and 2) privacy accounting using the MA method underestimates the actual privacy loss of their neural network training, because it simply regards random reshuffling as random sampling with replacement. To address these problems, we develop different privacy accounting methods for each of the batching methods, and our algorithm makes proper choices depending on which method is used for data batching. For privacy accounting under random sampling based batching, we show that CDP is unable to capture the privacy amplification effect of random sampling. To address that, we propose a relaxation of zCDP and convert it to (ϵ, δ)-DP. Compared with MA, which needs to compute log moments of privacy loss over a range of moment orders, our method uses explicit expressions to compute privacy loss and thus is more efficient, particularly when the noise scale of the Gaussian mechanism dynamically changes along the training.

In our approach, dynamic privacy budget allocation is applied to DP-SGD to improve the model accuracy. In model publishing the privacy loss of each learning step is not our primary concern. This allows us to allocate different privacy budgets to different training epochs as long as we maintain the same overall privacy guarantee. Our dynamic budget allocation approach is in contrast to the previous work [7], which employs a uniform privacy budget allocation, and uses the same noise scale in each step of the whole training process. Our dynamic privacy budget allocation approach leverages several different ways to adjust the noise scale. Our experimental results demonstrate that this approach achieves better model accuracy while retaining the same privacy guarantee.

Algorithm 1 presents our DP-SGD algorithm. In each iteration, a batch of examples is sampled from the training dataset and the algorithm computes the gradient of the loss on the examples in the batch and uses the average in the gradient descent step. The gradient clipping bounds per-example gradients by l_2 norm clipping with a threshold C. The Gaussian mechanism adds random noise $\mathcal{N}(0, \sigma_t^2 C^2 \mathbb{I})$ to $\sum_i \hat{\mathbf{g}}_t(x_i)$ to perturb the gradients in every iteration. We have a total privacy budget ρ_{total} and cumulative privacy cost c_t^{priv}. The way to update c_t^{priv} depends on which batching method is used. If the privacy cost exceeds the total budget, then the training is terminated. In the pseudo code, the function $AdpBudgetAlloc(\rho_{total}, t, T, schedule)$ is used to obtain the noise scale σ_t for the current training step t, according to $schedule$ that decides how the noise scale is adjusted during the training time.

IV. DETAILS OF OUR APPROACH

In this section, we present the details of our approach for differentially private deep learning. We first propose our dynamic privacy budget allocation techniques and then develop privacy accounting methods based on zCDP for different data batching methods.

Algorithm 1: Differentially Private SGD Algorithm

Input: Training examples $\{x_1, \ldots, x_N\}$, learning rate η_t, group size L, gradient norm bound C, total privacy budget ρ_{total}

1 Initialize w_0 ;
2 Initialize cumulative privacy loss $c_t^{priv} = 0$;
3 **for** $t = 1 : T$ **do**
4   **Dynamic privacy budget allocation**:
5     $\sigma_t \leftarrow AdpBudgetAlloc(\rho_{total}, t, T, schedule)$;
6   update c_t^{priv} according to data batching method, t and σ_t;
7   If $c_t^{priv} > \rho_{total}$, break ;
8   **data batching**:
9   Take a batch of data samples $\mathbb{B}_t$ from the training dataset;
10     $B = |\mathbb{B}_t|$;
11   **Compute gradient**:
12   For each $i \in \mathbb{B}_t$, $\mathbf{g}_t(x_i) \leftarrow \nabla_{w_t} \mathcal{L}(w_t, x_i)$;
13   **Clip gradient**:
14     $\hat{\mathbf{g}}_t(x_i) \leftarrow \mathbf{g}_t(x_i)/max\left(1, \frac{||\mathbf{g}_t(x_i)||_2}{C}\right)$;
15   **Add noise**:
16     $\widetilde{\mathbf{g}}_t \leftarrow \frac{1}{B}\left(\sum_i \hat{\mathbf{g}}_t(x_i) + \mathcal{N}(0, \sigma_t^2 C^2 \mathbb{I})\right)$;
17   **Descent**:
18     $w_{t+1} \leftarrow w_t - \eta_t \widetilde{\mathbf{g}}_t$;
19 Output w_T ;

A. Dynamic Privacy Budget Allocation

In Algorithm 1, the privacy budget allocated to an epoch decides the noise scale of Gaussian mechanism used by each iteration within that epoch. For a given privacy budget ρ_{total}, the final model accuracy depends on how the privacy budget is distributed over the training epochs. Our approach aims to optimize the budget allocation over the training process to obtain a differentially private DNN model with better accuracy.

Concretely, our dynamic privacy budget allocation follows the idea that, as the model accuracy converges, it is expected to have less noise on the gradients, which allows the learning process to get closer to the local optimal spot and achieve better accuracy. A similar strategy has been applied to the learning rate of DNNs in common practice. It is often recommended to reduce the learning rate as the training progresses, instead of using a constant learning rate throughout all epochs, to achieve better accuracy [3], [12]. Therefore, we propose a set of methods for privacy budget allocation, which effectively improve the model accuracy by dynamically reducing the noise scale over the training time (as demonstrated by our experiments).

1) Adaptive schedule based on public validation dataset: One approach for adjusting the noise scale is to monitor the validation error during training and reduce the noise scale whenever the validation error stops improving. We propose an adaptive privacy budget allocation that dynamically reduces the noise scale according to the validation accuracy. Every time when the validation accuracy improves by less than a threshold δ, the noise scale is reduced by a factor

of k until the total privacy budget runs out. However, when the validation dataset is sampled from the private training dataset, the schedule has some dependency on the private dataset, which adds to the privacy cost. In this case, if we can leverage a small publicly available dataset from the same distribution and use it as our validation dataset, then it will not incur any additional privacy loss.

In our approach, with a public validation dataset, the validation accuracy is checked periodically during the training process to determine if the noise scale needs to be reduced for subsequent epochs. The epochs over which the validation is performed are referred to as validation epochs. Let σ_e be the noise scale for the DP-SGD training in the validation epoch e, and S_e be the corresponding validation accuracy . The noise scale for the subsequent epochs is adjusted based on the accuracy difference between current epoch e and the previous validation epoch $e-1$. Initially, $S_0 = 0$,

$$\sigma'_e = \begin{cases} k\sigma_e, & \text{if } S_e - S_{e-1} \leq \delta & (8) \\ \sigma_e & (9) \end{cases}$$

The updated noise scale σ'_e is then applied to the training until the next validation epoch $e+1$. The above equations amount to say that if the improvement on the validation accuracy $S_e - S_{e-1}$ is less than the threshold δ, it triggers the decay of noise scale $\sigma_e = k\sigma_e$ where k $(0 < k < 1)$ is decay rate, a hyperparameter for the schedule. We note that the validation accuracy may not increase monotonically as the training progresses, and its fluctuations may cause unnecessary reduction of noise scale and thus waste on the privacy budget. This motivates us to use the moving average of validation accuracy to improve the effectiveness of validation-based noise scale adjustment: at validation epoch e, we define an averaged validation accuracy $\bar{S}_e$ over the previous m validation epochs from e, including itself, as follows:

$$\bar{S}_e = \frac{1}{m} \sum_{i=e-m+1}^{e} S_i \qquad (10)$$

The schedule checks the averaged validation accuracy every $period$ $(period \geq m)$ number of validation epochs and compares the current result with that of the last checking time to decide if the noise scale needs to be reduced according to Eq. (8).

2) Pre-defined schedules: When the public validation dataset is not available, we propose to use an alternative approach that pre-defines how the noise scale decreases over time without accessing any datasets or checking the model accuracy. Concretely, in our approach, the noise scale is reduced over time according to some decay functions. The decay functions update the noise scale by epoch, while the noise scale keeps the same for every iteration within an epoch. By leveraging the Gaussian mechanism that adds noise from $\mathcal{N}(0, \sigma_t^2 C^2 \mathbb{I})$, we provide four instances of decay functions, all of which take the epoch number t as an argument.

a) Time-Based Decay: It is defined with the mathematical form

$$\sigma_t = \sigma_0/(1 + kt)$$

where σ_0 is the initial noise parameter for σ, t is the epoch number and k $(k > 0)$ is decay rate. when $k < 1$, it is known as "search-then-converge" [13], which decreases the noise scale linearly during the SGD search phase when t is less than the "search time" $1/k$, and decreases the noise by $1/t$ when t is greater than $1/k$.
b) Exponential Decay: It has the mathematical form

$$\sigma_t = \sigma_0 e^{-kt}$$

where k $(k > 0)$ is decay rate.
c) Step Decay: Step decay reduces the learning rate by some factor every few epochs. The mathematical form is

$$\sigma_t = \sigma_0 * k^{\lfloor t/period \rfloor}$$

where k $(1 > k > 0)$ is the decay factor and $period$ decides how often to reduce noise in terms of the number of epochs.
d) Polynomial Decay: It has the mathematical form

$$\sigma_t = (\sigma_0 - \sigma_{end}) * (1 - t/period)^k + \sigma_{end}$$

where $k(k > 0)$ is the decay power and $t < period$. A polynomial decay function is applied to the initial σ_0 within the given number of epochs defined by $period$ to reach σ_{end} $(\sigma_{end} < \sigma_0)$. When $k = 1$, this is a linear decay function.

The schedules for noise decay are not limited to the above four instances. In this paper we choose to use these four because they are simple, representative, and also used by learning rate decay in the performance tuning of DNNs. Users can apply them in various ways. For example, a user can use them in the middle of training phase, but keep constant noise scale before and after. Note that in the time-based decay and exponential decay, t can be replaced by $\lfloor t/period \rfloor$ as done in the step decay such that the decaying is applied every $period$ number of epochs. The polynomial decay requires a specific end noise scale after $period$ epochs, and we make the noise scale constant after the training time exceed $period$ epochs.

3) Privacy Preserving Parameter Selection: The proposed schedules require a set of pre-defined hyperparameters, such as decay rate and period. Their values decide the training time and affect the final model accuracy. It is expected to find the optimal hyperparameters for the schedules to produce the most accurate model. A straightforward approach is to test a list of k candidates by training k neural networks respectively and trivially choose the one that achieves the highest accuracy, though this adds the privacy cost up to $k\rho_{total}$. A better approach is to apply differentially private parameter tuning, such as the mechanism proposed by Gupta et al. [11]. The idea is to partition the dataset to $k+1$ equal portions, train k models with using k schedules on k different data portions respectively, and evaluate the number of incorrect predictions for each model, denoted

by z_i $(1 \leq i \leq k)$, on the remaining data portion. Then, the Exponential Mechanism [29] is applied, which selects and outputs a candidate with the probability proportional to $exp(\frac{-\epsilon z_i}{2})$. This parameter tuning procedure satisfies ϵ-DP, and accordingly satisfies $\frac{1}{2}\epsilon^2$-zCDP [9].

B. Refined Privacy Accountant

The composition property of zCDP allows us to easily compute cumulative privacy loss for the iterative SGD training algorithm. Suppose that each iteration satisfies ρ-zCDP and the training runs T iterations, then the whole training process satisfies $(T\rho)$-zCDP. In this section we show that 1) the composition can be further refined by considering the property of the mini-batch SGD algorithm, and 2) more importantly, different batching methods lead to different privacy loss. In particular, we analyze the privacy loss composition under two common batching methods: random sampling with replacement and random reshuffling. With random reshuffling, the training dataset is randomly shuffled and then partitioned into batches of similar size and SGD sequentially processes one batch at a time. It is a random sampling process without replacement. For random sampling with replacement, each example in a batch is independently sampled from the training dataset with replacement. Because these two data batching methods have different privacy guarantees, for tracking privacy loss correctly, it is important for the users to choose the right accounting method based on the batching method they use.

1) Under random reshuffling: SGD takes disjoint data batches as input within an epoch with random reshuffling. We note that the existing results [9], [15], [17] on the composition of a sequence of differential private mechanisms $\mathcal{A}_1, \ldots, \mathcal{A}_k$ assume that each mechanism $\mathcal{A}_i$ runs with the same dataset X as input. It is expected that their composition has less cumulative privacy loss if each of differentially private mechanisms runs on disjoint datasets. The formal composition result in this scenario is detailed in Theorem 2. All the proofs of lemmas and theorems in this paper can be found in Appendix.

Theorem 2. *Suppose that a mechanism $\mathcal{A}$ consists of a sequence of k adaptive mechanisms, $\mathcal{A}_1, \ldots, \mathcal{A}_k$, where each $\mathcal{A}_i$: $\prod_{j=1}^{i-1} \mathcal{R}_j \times \mathcal{D} \to \mathcal{R}_i$ and $\mathcal{A}_i$ satisfies ρ_i-zCDP $(1 \leq i \leq k)$. Let $\mathbb{D}_1, \mathbb{D}_2, \ldots, \mathbb{D}_k$ be the result of a randomized partitioning of the input domain $\mathbb{D}$. The mechanism $\mathcal{A}(D) = (\mathcal{A}_1(D \cap \mathbb{D}_1), \ldots, \mathcal{A}_k(D \cap \mathbb{D}_k))$ satisfies $\frac{1}{k}(\sum_{j=1}^{k} \rho_i)$-zCDP.*

Theorem 2 provides a tighter characterization of privacy loss for the composition of mechanisms having disjoint input data. Assume for all $i \in (1, \ldots, k)$, $\rho_i = \rho$. Given Theorem 2, it is trivial to demonstrate that mechanism $\mathcal{A}$ satisfies ρ-zCDP. Compared with a guarantee of $(k\rho)$-zCDP using the composition property in Proposition 1, we demonstrate that the total privacy loss of sequential computations on

disjoint datasets is just ρ, equivalent with one computation step in the sequence. For ϵ-DP, the similar result in [30] of the parallel composition theorem says that when each ϵ_i-differentially private mechanism queries a disjoint subset of data in parallel and work independently, their composition provides $(\max_i \epsilon_i)$-DP instead of the $\sum_i \epsilon_i$ derived from a naive composition. One difference in our analysis is that we are able to use the mean of all ρ_i's as the final total privacy loss due to the random partitioning of the dataset.

In the differentially private mini-batch SGD algorithm shown in Algorithm 1, each iteration step t satisfies $(\frac{1}{2\sigma_t^2})$-zCDP according to Proposition 2. Suppose that each iteration from the same epoch uses the same noise scale σ and each uses a disjoint data batch. Then, by Theorem 2, we know that the computation of this epoch still satisfies $(\frac{1}{2\sigma^2})$-zCDP. Because the training dataset is repeatedly used every epoch, the composition of the epoch level computations follows normal composition of Proposition 1. Thus, when the training runs a total of E epochs and each epoch e satisfies ρ_e-zCDP, the whole training procedure satisfies $(\sum_{e=1}^{E} \rho_e)$-zCDP.

2) Under random sampling with replacement: We have shown a privacy amplification effect resulting from the disjoint data access of every iteration within one epoch under random reshuffling. In contrast, the MA method [7] exploits the privacy amplification effect of random sampling with replacement. In this section, we examine how random sampling with replacement affects the privacy loss in terms of zCDP. Intuitively, the random sampling with replacement introduces more uncertainty than the random reshuffling process which samples data batches without replacement. However, our analysis shows that CDP cannot characterize the privacy amplification effect of random sampling. It is because of the restrictive notion of sub-Gaussianity in CDP which requires moment constraints on all orders, i.e., $\alpha \in (1, \infty)$ in Eq. (7). To address this problem, we propose a relaxation of zCDP and convert it to (ϵ, δ)-DP. This then allows us to capture the privacy amplification of random sampling.

Suppose a new mechanism $\mathcal{A}'$ that runs ρ-zCDP mechanism $\mathcal{A}$ on a random subsample of dataset $\mathbb{D}$ where each example is independently sampled with probability q. Without loss of generality, we fix $\mathbb{D}$ and consider a neighboring dataset $\mathbb{D}' = \mathbb{D} \cup d_e$. we use $\Lambda(*)$ to denote the sampling process over dataset $*$, let T be any subsample that does not include d_e and $T' = T \cup d_e$. Because d_e is randomly sampled with probability q, $\mathcal{A}'(\mathbb{D}')$ is distributed identically as $u_0 \triangleq \Pr(\Lambda(\mathbb{D}) = T)\mathcal{A}(T)$ with probability $(1 - q)$, and as $u_1 \triangleq \Pr(\Lambda(\mathbb{D}') = T'|d_e \in T')\mathcal{A}(T')$ with probability q. Therefore, the following holds:

$$\mathcal{A}'(\mathbb{D}) \sim u_0, \qquad \mathcal{A}'(\mathbb{D}') \sim qu_1 + (1-q)u_0 \qquad (11)$$

By $\Pr(\Lambda(\mathbb{D}) = T) = \Pr(\Lambda(\mathbb{D}') = T'|d_e \in T')$ due to $T' =$

$T \cup d_e$, it is easy to prove that $\mathcal{A}'$ still satisfies ρ-zCDP. The proof details can be found in the Appendix D.

No privacy amplification for $\mathcal{A}'$ in terms of CDP. Consider

$$(\alpha - 1)D_\alpha(\mathcal{A}'(\mathbb{D}')||\mathcal{A}'(\mathbb{D}))$$
$$= \log\left(\mathbb{E}_{u_0}\left[(\frac{qu_1 + (1-q)u_0}{u_0})^\alpha\right]\right) > \log\left(\mathbb{E}_{u_0}\left[(q\frac{u_1}{u_0})^\alpha\right]\right)$$
$$= \alpha \log q + (\alpha - 1)D_\alpha(u_1||u_0) \tag{12}$$

When $\alpha \to \infty$, we have

$$D_\alpha(\mathcal{A}'(\mathbb{D}')||\mathcal{A}'(\mathbb{D})) > \alpha\rho + \frac{\alpha}{\alpha - 1}\log q \to \alpha\rho + \log q \tag{13}$$

This shows that the sampling does not produce any reduction with regard to q on ρ and still $D_\alpha(\mathcal{A}'(\mathbb{D}')||\mathcal{A}'(\mathbb{D})) = \Theta(\alpha\rho)$. Therefore, by definition, zCDP is not able to capture the privacy amplification effect of random sampling.

The reason there is no privacy amplification with random sampling on zCDP is that concentrated DP requires a sub-Gaussian distribution for privacy loss, and thus moments must be bounded by $\exp(O(\alpha^2))$ at all orders $\alpha \in (1, +\infty)$. This is a fairly strong condition. Alternatively, it requires that a mechanism $\mathcal{A}$ has α-Rényi divergence bounded by $\alpha\rho$ for all $\alpha \in (1, +\infty)$. To demonstrate that, we assume a Gaussian mechanism $\mathcal{A}$ that $\mathcal{A}(T) \sim N(0, \sigma^2)$ and $\mathcal{A}(T') \sim N(1, \sigma^2)$. Denote $N(0, \sigma^2)$ by p_0 and $N(1, \sigma^2)$ by p_1. It is trivial to show that $D_\alpha(u_1||u_0) = D_\alpha(p_1||p_0)$ and $D_\alpha(qu_1 + (1-q)u_0||u_0) = D_\alpha(qp_1 + (1-q)p_0||p_0)$. We can then numerically compute $D_\alpha(u_1||u_0)$ and $D_\alpha(qu_1 + (1-q)u_0||u_0)$ with $q = 0.01$ and $\sigma = 4$. Results are shown in Figure 1. $D_\alpha(u_1||u_0)$ is linear in α because $D_\alpha(u_1||u_0) = \alpha\rho = \alpha/(2\sigma^2)$. However, we can see that $D_\alpha(qu_1 + (1-q)u_0||u_0)$ has a changing point at $\alpha = 147$. Before this changing point it is close to zero, because $q = 0.01$ is very small and the two distributions $qu_1 + (1-q)u_0$ and u_0 are close to each other. At higher orders, the sampling effect vanishes and the divergence increases at the rate of $D_\alpha(u_1||u_0)$. This indicates that the privacy amplification effect from random sampling does not hold for all orders $\alpha \in (1, +\infty)$, and we cannot improve the privacy metric in terms of zCDP under the random sampling. On the other hand, Figure 1 suggests that we need to analyze the α-Rényi divergence within a limited range of α to capture the privacy amplification effect. To analytically characterize this, we consider the Gaussian mechanism and derive the following result using the log moment bounds given in [7]:

Lemma 1. *Suppose that $f : \mathbb{D} \to \mathbb{R}^p$ with $||f(\cdot)||_2 \leq 1$. Consider a mechanism $\mathcal{A}'$ that runs a Gaussian mechanism adding noise $\mathcal{N}(0, \sigma^2\mathbb{I})$ over a random subsample $J \subseteq \mathbb{D}$ where each example is independently sampled with probability q, i.e., $\mathcal{A}'(\mathbb{D}) = \sum_{i \in J} f(x_i) + \mathcal{N}(0, \sigma^2\mathbb{I})$. Let $q \leq \frac{1}{16\sigma}$, then, for two neighboring datasets $\mathbb{D}$ and $\mathbb{D}'$, the α-Rényi*
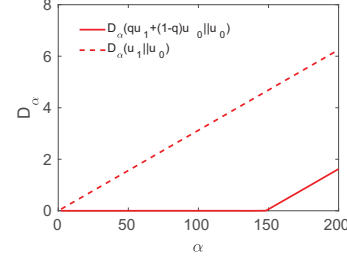

Figure 1: α-Rényi divergence under sampling(q=0.01, σ=4)

divergence between $\mathcal{A}'(\mathbb{D})$ and $\mathcal{A}'(\mathbb{D}')$ is given below:

$$D_\alpha(\cdot||\cdot) \leq q^2\alpha/\sigma^2, \quad \text{for } 1 < \alpha \leq \sigma^2 \log\frac{1}{q\sigma} + 1 \tag{14}$$

Eq. (14) defines a sub-exponential variable, a relaxation to sub-Gaussianity, by putting constraints on α. Such a relaxation is able to capture the privacy amplification effect of random sampling but it does not fit into the definition of CDP. Thus, we convert it to (ϵ, δ)-DP with the following theorem:

Theorem 3. *Let $\widehat{\rho} = q^2/\sigma^2$ in (14). If the mechanism $\mathcal{A}'$ has*

$$D_\alpha(\mathcal{A}'(\mathbb{D})||\mathcal{A}'(\mathbb{D}')) \leq \alpha\widehat{\rho} \tag{15}$$

for $1 < \alpha \leq \sigma^2 \log\frac{1}{q\sigma} + 1$, it satisfies

$$\begin{cases} \left(\widehat{\rho} + 2\sqrt{\widehat{\rho}\log(1/\delta)}, \delta\right)\text{-DP, if } \delta \geq 1/\exp\left(\widehat{\rho}\sigma^4(\log\frac{1}{q\sigma})^2\right) & (16) \\ \left(\widehat{\rho}(\sigma^2 \log\frac{1}{q\sigma} + 1) - \frac{\log\delta}{\sigma^2\log(1/q\sigma)}, \delta\right)\text{-DP, otherwise} & (17) \end{cases}$$

Composition Theorem 3 captures the privacy amplification of random sampling with replacement by converting the bound of α-Rényi divergence within a limited range of α to (ϵ, δ)-DP. Following this, we consider the composition of a sequence of Gaussian mechanisms with random sampling. Suppose k mechanisms, denoted by $\mathcal{M} = (\mathcal{A}'_1, \ldots, \mathcal{A}'_k)$ where each $\mathcal{A}'_i$ uses sampling ratio q_i and noise scale σ_i. Because the constraint of α in Eq. (14) depends on the sampling ratio and noise scale, we examine their composition in two cases:

1.) Each mechanism uses the same q and σ. For $1 < \alpha \leq \sigma^2 \log\frac{1}{q\sigma} + 1$, by the composition property of α-Rényi divergence [9], we have $D_\alpha(M(\mathbb{D})||M(\mathbb{D}')) \leq kq^2\alpha/\sigma^2$. The conversion to (ϵ, δ)-DP can be done by letting $\widehat{\rho} = kq^2/\sigma^2$ in (16) and (17) in Theorem 3.

2.) The sampling ratio and noise scale are different for each mechanism. Then, for each $\mathcal{A}_i$, we have $D_\alpha(\mathcal{A}_i(\mathbb{D})||\mathcal{A}_i(\mathbb{D}')) \leq q_i^2\alpha/\sigma_i^2$ for $1 < \alpha \leq \sigma_i^2 \log\frac{1}{q_i\sigma_i} + 1$. To apply Lemma 1 and allow the composition of α-Rényi divergence of mechanisms with different q_i and σ_i, we constrain α to the range $1 < \alpha \leq \min_i\{\sigma_i^2 \log\frac{1}{q_i\sigma_i}\} + 1$. It is then clear that $D_\alpha(M(\mathbb{D})||M(\mathbb{D}')) \leq \alpha(\sum_i \frac{q_i^2}{\sigma_i^2})$ holds within this α range. Letting $\widehat{\rho} = \sum_i \frac{q_i^2}{\sigma_i^2}$ and replacing q and σ by q_j and σ_j where $j = \arg\min_i\{\sigma_i^2 \log\frac{1}{q_i\sigma_i}|1 \leq i \leq k\}$

in Eq. (16) and (17), we can still obtain the corresponding (ϵ,δ)-DP.

When random sampling is used for batching, Algorithm 1 follows the above method to estimate privacy loss in terms of (ϵ,δ)-DP. In particular, the algorithm specifies a fixed $\delta = \delta_0$ and a total privacy budget ϵ_{total}, and at every iteration step t, it updates $\widehat{\rho} = \sum_{i=0}^{t} \frac{q_i^2}{\sigma_i^2}$ and computes the corresponding cumulative privacy loss ϵ_t. If $\epsilon_t > \epsilon_{total}$, the training is terminated and the final model satisfies $(\epsilon_{total}, \delta_0)$-DP.

In summary, we have shown that CDP is not able to capture the privacy amplification effect of random sampling. We address this issue by bounding α-Rényi divergence over a constrained range of α instead of $(1,\infty)$ and convert to (ϵ,δ)-DP. On one hand, our approach is similar to the previous MA approach [7] and it produces very close results compared to that of MA with regard to (ϵ,δ)-DP guarantee, as shown in our experiment in Section V-A. On the other hand, however, our approach starts from a different viewpoint and its computation by following Eq. (16) and (17) is much simpler than MA which needs to compute log moments of privacy loss variable for a range of α. At the same time, our privacy accounting approach allows each iteration to use different sampling ratios and noise scales and is more efficient to compute than MA when applied with dynamic privacy budget allocation.

More importantly, we have provided formal analysis to show that the compositions of differential privacy under two batching methods are distinct. As demonstrated by our experimental results, this causes different privacy loss. Therefore, we argue that the privacy accounting method has to be chosen according to which data batching method is used. In our implementation, we focus on random reshuffling, because it is a common practice in the neural network implementation [5], [22]. In fact, several existing deep learning frameworks such as TensorFlow provide convenient random reshuffling APIs for generating batches. It is also numerically observed that random reshuffling outperforms its random sampling with replacement [23].

C. DP Composition Under Dynamic Schedules

For pre-defined schedules, once the hyperparameters are specified, they follow the decay functions to update the noise scale without accessing the data and the model, and thus do not incur any additional privacy cost. Since the noise scale is updated by epoch, each iteration step within an epoch uses the same noise scale. Suppose the epoch t uses the noise scale σ_t. Each iteration of epoch t is then $\rho = 1/(2\sigma_t^2)$ zCDP by Proposition 2, and the total privacy cost of epoch t can be calculated by Theorem 2 or 3 depending on which batching method is used. Over the course of training, the cumulative privacy loss is updated at each epoch, and once the cumulative privacy loss exceeds the fixed privacy budget ρ_{total}, the training is terminated. To achieve a target training time under a given total privacy budget, we can determine

the exact values of hyperparameters for these schedules in advance before the training time.

For the validation-based schedule, the access to the public validation dataset does not incur additional privacy cost. With this schedule, the composition of differential privacy involves adaptive choices of the privacy parameter ρ at every epoch, which is corresponding to the noise scale σ of the Gaussian mechanism. This means that the choice of privacy parameters itself is a function of the realized outcomes of the previous rounds. It has been shown by Rogers et al. [31] that the strong composition theorem for (ϵ,δ)-DP fails to hold in this adaptive privacy parameter setting since the theorem requires the privacy parameters to be pre-defined ahead of time. To address this problem, they define the privacy loss as a random variable as done in Eq. (5) for CDP and develop the composition for (ϵ,δ)-DP using privacy filters. Privacy filters provide a way to halt the computation with probability $1-\delta$ before the realized privacy loss exceeds ϵ. Our approach relies on zCDP which defines privacy loss as in Eq. (5) by nature and therefore the composition accumulating privacy cost with regard to Rényi divergence holds for the adaptive parameter settings.

V. Experimental Results

In this section, we evaluate the proposed privacy accounting methods, and demonstrate the effectiveness of dynamic privacy budget allocation on different learning tasks. Our implementation is based on the TensorFlow implementation [1] of DP-SGD in the paper [7].

A. Comparing Privacy Accounting Approaches

In Section IV-B we derive different privacy accounting methods for two data batching methods: random reshuffling (RF) and random sampling with replacement (RS). We refer to them as zCDP(RF) and zCDP(RS) respectively. To numerically compare them with other privacy accounting methods including strong composition [18] and the moments accountant (MA) method [7], we unify them into (ϵ,δ)-DP. Following [7], we assume that the batches are generated with RS for both the strong composition and MA. We use the implementation of [7] in TensorFlow to compute MA. For strong composition, we apply the strong composition theorem in [18] to the composition of $(\log(1+q(e^\epsilon-1)), q\delta)$-DP mechanisms that are the privacy amplified version of (ϵ,δ)-DP mechanisms running with random sampling ratio q. We compute (ϵ,δ)-DP for zCDP(RF) with Proposition 1 and for zCDP(RS) with the methods in Section IV-B2.

In our experimental setting, we assume a batch size B for random reshuffling. For random sampling we assume a sampling ratio $q = \frac{B}{N}$ given a total of N samples. In the following, when we vary q, it is equivalent to change the batch size for random reshuffling. The number of iterations in one epoch is $\frac{1}{q}$. For simplicity, we use the same noise scale for Gaussian mechanism $\mathcal{N}(0, C^2\sigma^2\mathbf{I})$ for every iteration

and set $q = 0.01$ and $\sigma = 6$ by default. Given $\sigma = 6$, the Gaussian mechanism satisfies both ($\epsilon = 0.808$, $\delta = 1e - 5$)-DP and $\rho = 0.0139$-zCDP. We track the cumulative privacy loss by epoch with different privacy accounting methods and convert the results to ϵ in terms of (ϵ, δ)-DP with fixed $\delta = 1e - 5$.

Figure 2 shows the growth of privacy loss metric ϵ during the training process. It shows that zCDP(RF) has lower estimation on privacy loss than that of the strong composition during the training. The final spent ϵ at epoch 400 by zCDP(RF) and strong composition are 21.5 and 34.3 respectively. Although random sampling introduces higher uncertainty and thus less privacy loss than random shuffling, zCDP(RF) still achieves lower and thus tighter privacy loss estimation even than the strong composition with random sampling. This demonstrates the benefit of CDP for composition of a large number of computations. The results for zCDP(RS) and MA are very close to each other because that they both exploit the moment bounds of privacy loss to achieve tighter tail bound and take advantage of the privacy amplification of random sampling. The final spent ϵ is 2.37 and 1.67 for zCDP(RS) and MA respectively. The reason for zCDP(RS) to have a slightly higher estimation is that its conversion to (ϵ, δ)-DP explicitly uses the log moment bound instead of the numerical computation of log moments. The benefit of zCDP(RS) is that it is easy to compute with explicit expressions in Theorem 3, and its composition for dynamic privacy budget allocation is also simple and thus more efficient.

Figure 2 shows that zCDP(RF) has higher privacy loss compared to MA and zCDP(RS), because more uncertainty is introduced with RS. However, it is worth noting that the common practice in deep learning is to use RF, including the implementation of [7]. Thus, zCDP(RF) is the proper choice for them and also straightforward due to the composition property of ρ-zCDP which simply adds up on ρ values. The results show that MA underestimates the real privacy loss when treating the random reshuffling as random sampling with replacement.

We further examine how zCDP(RF) and zCDP(RS) change with the sampling ratio q and the noise scale σ. Using the default $\sigma = 6$, Figure 3a shows the privacy loss ϵ at the end of 200 training epochs with varying q values. For zCDP(RF), the cumulative privacy loss does not change with q. This is because the composition of ρ-zCDP iterations within one epoch still satisfies ρ-zCDP by Theorem 2 and across epochs the linear composition of ρ-zCDP in Theorem 1 is applied, which makes the final privacy loss depend exclusively on the number of training epochs. We have fixed 200 epochs so the final privacy loss does not change. In contrast, the privacy loss given by zCDP(RS) increases with the sampling ratio q, which can be seen in Eq. (16) where ϵ increases with $\widehat{\rho}$ which is proportional to q^2. Similarly, Figure 3b shows the privacy loss after
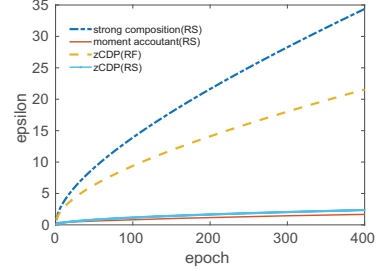

Figure 2: Privacy parameter ϵ v.s. epoch

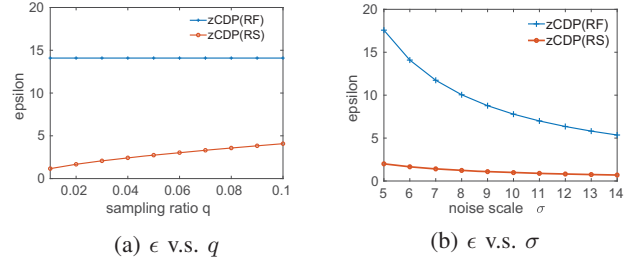(a) ϵ v.s. q (b) ϵ v.s. σ

Figure 3: privacy loss ϵ v.s. sampling ratio q & noise scale σ

200 epochs by varying noise scales with the same $q=0.01$. We observe that increasing σ from 5 to 14 significantly reduces ϵ for zCDP(RF) but has noticeably less impact on zCDP(RS). It suggests that under random sampling, a small sampling ratio contributes much more on privacy than the noise scale σ. This indicates that we may reduce the noise scale to improve the model accuracy without degrading much privacy. However, for random reshuffling, the privacy loss does not depend on the sampling ratio (i.e., the batch size) but is decided by σ, so it is more critical to achieve a good trade-off between privacy and model accuracy in this case. Our privacy budget allocation techniques optimize this trade-off by dynamically adjusting σ during the training to improve model accuracy while retaining the same privacy guarantee.

B. Evaluating Dynamic Privacy Budget Allocation

In this section we evaluate the effectiveness of dynamic privacy budget allocation compared to uniform privacy budget allocation adopted by Abadi et al. [7]. Since the TensorFlow implementation uses random reshuffling to generate batches, privacy accounting in Section IV-B1 should be used to avoid the underestimation of privacy loss. We therefore use ρ as the metric to represent the privacy budget and loss. Because the techniques for adjusting noise scales are independent of the batching method, the benefit of dynamic privacy budget allocation on model accuracy demonstrated under random reshuffling also applies to random sampling.

1) Datasets and Models: Our experiments use three datasets and different default neural networks for each dataset.

MNIST. This is a dataset of handwritten digits consists of 60,000 training examples and 10,000 testing examples [26]

Table I: Budget Allocation Schedules under Fixed Budget $\rho_{total} = 0.78125$, initial noise scale $\sigma_0 = 10$ for dynamic schedules

	Uniform [7] (σ_c =8)	Time (k=0.05)	Step(k=0.6, $period$=10)	Exp (k=0.01)	Poly(k=3,σ_{end}=2, $period$=100)	Validation(k=0.7,m=5, δ=0.01,$period$=10)
epochs	100	38	31	71	44	64
training accuracy	0.918	0.934	0.928	0.934	0.930	0.930
testing accuracy	0.919	0.931	0.929	0.929	0.932	0.930
non-private SGD	0.978/0.970	0.959/0.957	0.955/0.954	0.971/0.965	0.963/0.959	0.97/0.964
uniform/same #epochs		0.922/0.925	0.921/0.925	0.922/0.925	0.925/0.929	0.924/0.926

formatted as 28X28 size gray-level images. In our experiment, the neural network model for MNIST follows the settings in previous work [7] for comparison: a 60-dimensional PCA projection layer followed by a simple feed-forward neural network comprising a single hidden layer of 1000 ReLU units. The output layer is softmax of 10 classes corresponding to the 10 digits. The loss function computes cross-entropy loss. A batch size 600 is used. The non-private training of this model can achieve 0.98 accuracy with 100 epochs.

Cancer Dataset. This dataset [2] consists of 699 patient examples. Each example has 11 attributes including an id number, a class label that corresponds to the type of breast cancer (benign or malignant), and the 9 features describing breast fine-needle aspirates. After excluding 16 examples with missing values, we use 560 examples for training and 123 examples for testing. A neural network classifier with 3 hidden layers, containing 10, 20, and 10 ReLU units, is trained to predict whether a breast tumor is malignant or benign. Each iteration takes the whole training data set as a batch and thus each iteration is one epoch. The non-private training of this model achieves testing and training accuracy 0.96 with 800 epochs.

CIFAR-10. The CIFA-10 dataset consists of 32×32 color images with three channels (RGB) in 10 classes including ships, planes, dogs and cats. Each class has 6000 images. There are 40,000 examples for training, 10,000 for testing and 10,000 for validation. For experiments on CIFAR, we use a pre-trained VGG16 neural network model [35]. Following the previous work [7], we assume the non-private convolutions layers that are trained over a public dataset (ImageNet [4] for VGG16) and only retrain a hidden layer with 1000 units and a softmax layer with differential privacy. We use 200 training epochs and batch size of 200. The corresponding non-private training achieves 0.64 training accuracy and 0.58 testing accuracy.

2) Results on MNIST: In differentially private model training, we keep the batch size at 600, clip the gradient norm of every layer at 4, and use fixed noise scale $\sigma_{pca} = 16$ for differentially private PCA. Note that, since the PCA part has constant privacy cost $1/(2\sigma_{pca}^2)$ in terms of ρ-zCDP, we exclude it from the total privacy budget ρ_{total} in our experiment, i.e., ρ_{total} is only for the DP-SGD in Algorithm 1. A constant learning rate 0.05 is used by default. We evaluate the model accuracy during training under different privacy budget allocation schedules. For the validation-based schedule, we divide the training dataset into 55,000 examples for training and 5000 examples for validation, and perform validation every epoch.

The results in Table I demonstrate the benefit of dynamic privacy budget allocations and the effect of earlier training termination on the model accuracy. For comparison, we consider the uniform privacy budget allocation in [7] with a constant noise scale σ_c=8 for every epoch as our baseline. We choose a fixed total privacy budget ρ_{total}=0.78125. This results in 100 training epochs in the baseline case. We test all dynamic schedules with the hyperparameters given in the table and present their testing and training accuracy in numbers rounded to two decimals. The training is terminated when the privacy budget runs out, and the hyerparameters are chosen from a set of candidates to demonstrate varied training times in epochs which are reported in the table. We also ran a non-private version of SGD with using the same training time as these schedules to see the impact of DP-SGD on accuracy. We can see from Table I that the baseline with constant σ_c=8 achieves 0.918 training accuracy and 0.919 testing accuracy. By comparison, all non-uniform privacy budget allocation schedules improve the testing/training accuracy by 1%∼1.6% while running fewer epochs. Because DP-SGD is a randomized procedure and the numbers in the table vary among trials, we repeat all the experiments 10 times and report in Figure 5 the mean accuracy along with the min-max bar for every schedule. These results show that dynamic schedules consistently achieve higher accuracy than the baseline. Therefore, given a fixed privacy budget, the dynamic budget allocation can achieve better accuracy than using the uniform budget allocation.

The accuracy improvement shown with dynamic schedules in the above example comes from two sources: less training time and non-uniformity of budget allocation (i.e., decaying of the noise scale). With uniform allocation under a fixed privacy budget, reducing the training time increases the privacy budget allocated to every epoch and thus decreases the noise scale used by the Gaussian mechanism. Therefore, when the model training benefits more from the reduced perturbation rather than longer training time, using less training time can improve the model accuracy. As verification, we apply uniform budget allocation with the noise scale $\sqrt{T/(2\rho_{total})}$ to achieve the same training time T as the corresponding dynamic schedule. The training accuracy and
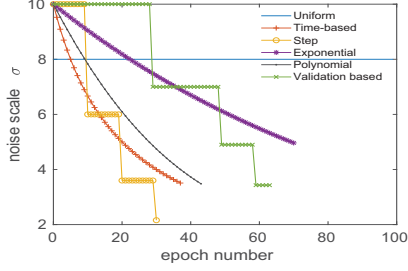
Figure 4: The change of noise scale σ during training

Figure 5: The accuracy comparison of different schedules

Figure 6: The accuracy under fixed training time

testing accuracy are presented respectively in the final row of Table I. All cases outperform the baseline case with $\sigma_c = 8$ with less than 100 training epochs, indicating the benefit of trading the training time for more privacy budget per epoch. however, it is worth noting that in certain cases, increasing the noise scale to prolong the training time may help improve the accuracy, exemplified by the result of the validation-based schedule on the Cancer dataset. Overall, when compared with the uniform allocation under the same training time, dynamic schedules demonstrate higher accuracy, therefore illustrating the benefit of non-uniformity and dynamic budget allocation.

Figure 4 shows how the noise scale σ changes with the epochs under different schedules in Table I. The curves terminate at the end of the training due to the depleted privacy budget. For the validation-based decay, the duration of the noise scale keeping unchanged decreases over the training time. The noise scale keeps 10 for 29 epochs, 7 for 20 epochs and 4.9 for 10 epochs. It is because that, as the model converges, the increment rate of the validation accuracy declines and it is more often to find that the accuracy increment does not exceed the given threshold.

We additionally manipulate different hyperparamters individually while keeping the rest constant to demonstrate their effects on training/testing accuracy and training time.. By default all accuracy numbers are the average of five trials. **The effect of decay functions.** In our previous experiments, we evaluate four types of decay functions for the pre-defined schedules. Here we compare their effects on the model accuracy with constant training time. Given the total privacy budget and initial noise scale, we can use the composition theorem of ρ-zCDP to search for proper values of the parameter k in these functions for the schedule to achieve a target training time. Table II in Appendix F provides the values of k for different decay functions to achieve training times of 60, 70, 80, 90, and 100 epochs respectively. These values are derived via search in step size $1e-4$, with the same initial noise scale and other parameters as stated in Table I.

Figure 6 shows the training and testing accuracy of pre-defined schedules under different training times along with the accuracy achieved by uniform budget allocation [7] using the same privacy budget. We observe that all training in-
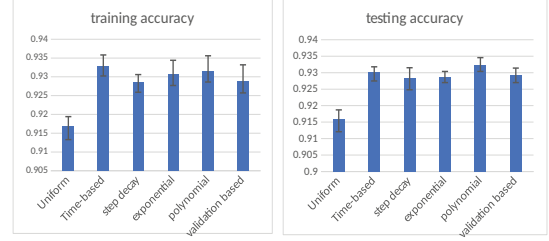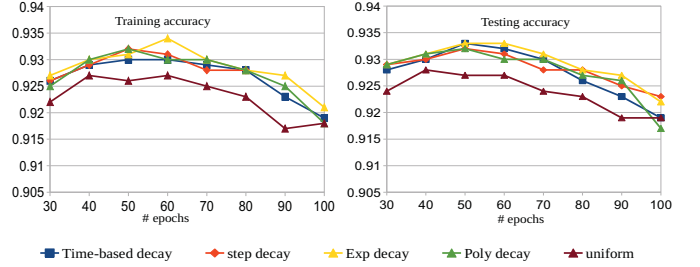
stances using pre-defined schedules achieve higher accuracy compared to the uniform budget allocation given a fixed training time. However, there is no clear winner among the different decay functions. Their accuracy increases from 30 to 50 or 60 epochs and then decreases as the training time increases from 50 or 60 to 100 epochs. At 100 epochs, all pre-defined schedules have an accuracy closer to that of the uniform budget allocation schedule running 100 epochs. Due to the similar behaviors of the different decay functions, we choose to simply set the decay function to the exponential decay in subsequent experiments unless otherwise stated.

Decay rate. The decay rate k decides how fast the noise scale decays. Keeping other parameters as the same as those reported in Table I, we vary k from 0.005 to 0.5 for exponential decay and from 0.3 to 0.9 for validation-based decay. Figure 7 and 8 show the accuracy and training time under different values of k. We make three observations. First, in both cases, there exists an optimal decay rate to achieve maximum accuracy. For exponential decay, the best accuracy occurs at $k=0.2$; for validation-based decay, it occurs at $k=0.7$. Second, for exponential decay, with the increase of k, the noise scale decreases at a higher rate. The privacy budget is therefore spent faster and the training time strictly decreases. For the validation-based decay, it reduces the noise scale to a k fraction of the original, so the decay rate is actually 1-k and the training time should increase with k. Figure 8 shows that the training time overall increases with k but with some random fluctuations. This is because that the validation-based decay adjusts the noise scale according to the validation accuracy which may change during the training in a non-deterministic way.

Third, the results at the ends of the x-axis in both figures indicate two interesting facts. At one end, although the lowest decay rate leads to the longest training time, it
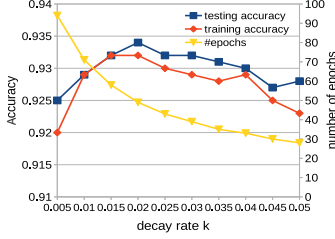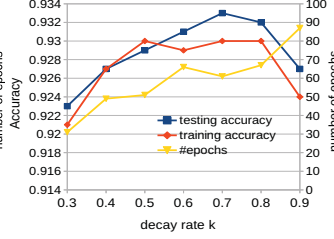
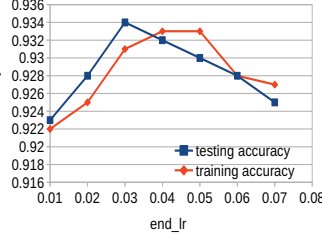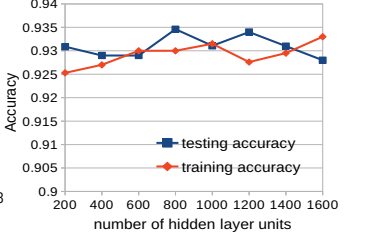Figure 7: exponential decay Figure 8: validation-based Figure 9: learning rate Figure 10: hidden units

also produces the worst accuracy because the noise scale decays more slowly in this case and therefore more epochs will suffer relatively higher noise scales. This degrades the efficiency of the learning process and lowers the accuracy. At the other end of the axis, the highest decay rate causes the training to stop much earlier, resulting in an insufficient training time which also degrades the accuracy.

Learning rate. Next we fix the decay rate k=0.0138 for exponential decay. Given a privacy budget 0.78125, the training lasts 60 epochs. With an initial learning rate 0.1, we linearly decrease the learning rate to end_{lr} over 10 epochs and then fix it at end_{lr} thereafter. We vary end_{lr} from 0.01 to 0.07. Figure 9 shows that the accuracy decreases significantly when the learning rate is too small or too large.

Number of hidden units/layers. Next, we vary the number of hidden units in the model from 200 to 1600. The results are shown in Figure 10. Although more hidden units increase the sensitivity of the gradient, leading to more noise added at each iteration, we observe that increasing the number of hidden units does not decease the model accuracy under the exponential decay schedule. This is consistent with the observation in [7] using uniform budget allocation. This shows that the effectiveness of dynamic budget allocation schedules scales to neural networks of different sizes. We also vary the number of hidden layers from 1 to 3, each with 1000 hidden units. The accuracy results are given in Appendix E and are consistent with [7] under the uniform allocation wherein the authors claim that, for MNIST, one hidden layer combined with PCA works better than networks with more layers.

Initial noise scale. In the previous experiments, the initial noise scale σ_0=10 is set as the default. To examine the effect of σ_0, we vary its value from 7 to 20 and measure the model accuracy under the fixed privacy budget 0.78125 in two cases: 1) for each σ_0, we choose the exponential decay rate to achieve a fixed training time of 60 epochs; 2) the exponential decay rate is fixed to 0.015, leading to variation in training time. Figure 11a and 11b show that, overall, increasing the initial noise scale reduces accuracy. In comparing the two figures, we observe that when the training time is fixed, the choice of σ_0 has less impact on accuracy. This is because, when the training time is fixed, a larger σ_0 results in a higher decay rate of the noise scale which benefits accuracy. However, for fixed decay rate, although

higher σ_0 leads to more training epochs, there is no accuracy improvement. This indicates that the model accuracy is more sensitive to the noise scale than the training time.

Accuracy and privacy in training. Figure 12 and 13 illustrate the change of model accuracy and privacy loss during training time for schedules with the same parameters as in Table I except the parameters explicitly noted in the figures. Figure 13 shows that the uniform privacy budget allocation in [7] incurs linear growth of privacy loss in terms of ρ-zCDP while our dynamic budget allocation schedules have faster growth rate due to the reduction of noise scale with time. All instances stop when the given total privacy budget of 0.78125 is reached. Combined with Figure 12, we can see that the exponential decay schedule consistently achieves better accuracy before the training ends compared to the uniform allocation, thanks to its faster noise scale reduction, while the validation-based schedule performs more conservatively and has a relatively longer training time.

An important implication of Figure 12 is that the gap between uniform budget allocation and non-private SGD indicates the maximum potential for the accuracy improvement through dynamic budget allocation over the uniform allocation. The proposed dynamic budget allocation schedules provide users a way to improve the accuracy of DP-SGD to approach that of non-private SGD. It is not possible for dynamic budget allocation to completely close this gap because gradient perturbation inevitably hurts model accuracy. Therefore, we argue that the effectiveness of dynamic privacy budget allocation should be evaluated on how much it can reduce the gap between non-private SGD and DP-SGD with uniform allocation. In our experiment, the accuracy difference between non-private SGD and DP-SGD of uniform case is 0.05 at the end of training. The exemplified schedules reduce this difference by 20%∼30%. One of our ongoing research directions is to investigate the ways to effectively find the best hyperparameters to apply these schedules.

3) Results on other datasets: We repeat the experiments on the Cancer Dataset and CIFAR-10 datasets. By applying exponential decay and validation-based decay to each learning task, we compare corresponding model accuracy with the uniform allocation method [7]. In this set of experiments, we first consider a uniform schedule that uses a constant noise scale to achieve a desired training time under the

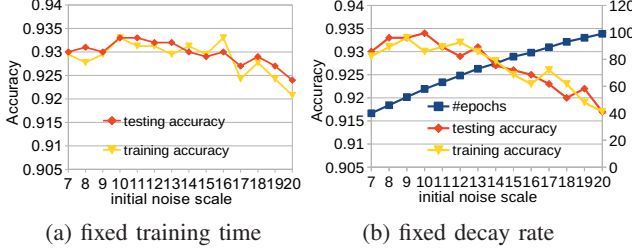(a) fixed training time   (b) fixed decay rate
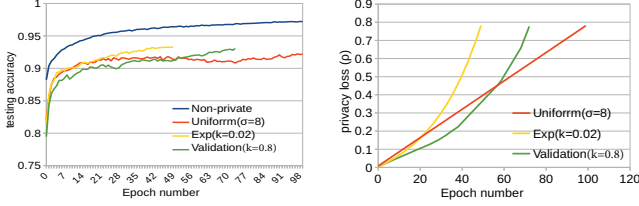
Figure 11: Initial noise scale

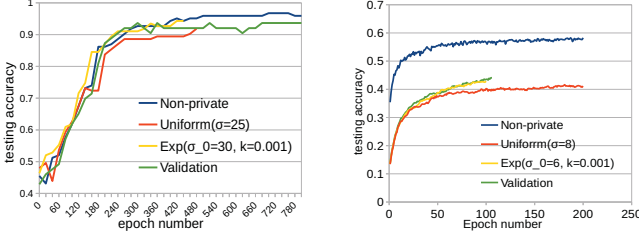Figure 12: Accuracy in training   Figure 13: Privacy in training

Figure 14: Accuracy (Cancer)   Figure 15: Accuracy (Cifar-10)

given privacy budget. Then we choose a value around this noise scale as the initial noise scale for decay schedules. A set of candidates for the decay rate is evaluated, and we use each candidate to train a model and compare achieved model accuracy. The parameters for the schedules we used are given in Table IV in Appendix G.

Results for testing and training accuracy are show in Figures 14 and 15. For the Cancer dataset, the exponential decay produces the model accuracy closer to the non-private SGD, about 3% higher accuracy than the uniform allocation case, and reduces the gap between non-private SGD and DP-SGD with uniform allocation by 70%. The validation-based schedule produces about 1.8% higher accuracy than the uniform case, with taking advantage of a longer training time as shown in Figure 14. For CIFAR-10, the exponential decay achieves 2% higher accuracy than the uniform case, and reduces the gap by about 12%. The validation-based schedule improves model accuracy by 4% over the uniform case, and reduces the gap by about 19%.

VI. DISCUSSION

We discuss a number of nuances/caveats as take-away remarks for deploying differentially private deep learning in practice for model publishing.

Understanding privacy parameter. Although differential privacy (DP) as a theory has evolved through different forms, today it is still not clear how a realistic privacy benefit

can be realized as a function of the privacy parameters in the DP definitions such as the ϵ and δ parameters in traditional DP and the ρ in zCDP. These privacy parameters lack understandable interpretations to the end-users. For ρ-zCDP, results like Proposition 1 would help if ϵ and δ had straightforward privacy-related interpretations. Advancement in interpretability and usability of DP parameters by end-users and domain-scientists can have profound impact on the practical deployment of differential privacy.

Data Dependency. The characteristics of input data, for example, dependency among training instances or dependency in the presence of training instances can render a differentially private mechanism ineffective for protecting the privacy of individuals [24], [25], [28]. The baseline definition of differential privacy is focused on the privacy of a single instance and therefore when multiple instances of the same user are present, a DP mechanism needs to be extended to group-level differential privacy to provide sufficient protection. One direction of our future work is to investigate and explore the ways of extending our DP-SGD techniques to provide a group-level privacy guarantee.

Resilience to Privacy Risks and Attacks. Differentially private deep learning aims to compute model parameters in a differentially private manner to limit the privacy risk associated with output model parameters. There are a number of known attacks in deep learning such as model inversion attacks and membership inference attacks. Model inversion attacks exploit the prediction output along with model access to infer an input instance. Membership inference attacks exploit the black box access to the prediction API to infer the membership of individual training instances. However, there is no formal study on whether or not a differentially private deep learning model is resilient to such attacks and what types of privacy risks known in practice can be protected with high certainty by a differentially private DNN model. In fact, DP only absolves the differentially-private release as a (quantifiably) strong cause of an inference. The work [21] provides an upper bound on the inferential privacy guarantee for differentially private mechanisms. DP, however, does not prevent the inference. This is another grand challenge in differential privacy and data privacy in general.

VII. RELATED WORK

Privacy threats in machine learning Existing works [19], [20], [34], [36] have shown that machine learning models and their usage may leak information about individuals in the training dataset and input data. Fredrikson, et al. [20] proposed a model inversion attack, which uses the output/prediction produced by a model to infer the unknown features of the input data and apply this attack against decision trees and neural networks in a pharmacogenetics scenario [19]. Reza et al. [34] developed a membership inference attack that aims to determine if an individual record was used as part of the training dataset for the

model using only the black-box access to the target model. Song et al. [36] proposed training phase attacks which perform minor modifications to training algorithms to make them output models which encode a significant amount of information about the training dataset while achieving high quality metrics like accuracy and generalizability. In addition, model extraction attacks proposed in [38] aim to duplicate the functionality of the model with black-box access. Such attacks can be leveraged to infer information about the model's training dataset.

Privacy-preserving deep learning To enable deep learning over the data from multiple parties while preserving the privacy of each party's training dataset, Reza et al [33] proposed a distributed deep learning framework that lets the participants train their model independently on their own dataset and only selectively share a subsets of their models' parameters during training. Abadi et al. [7] proposed a differentially private SGD algorithm for deep learning to offer provable privacy guarantees on the output model. DP [14] as a defacto standard for privacy has been applied to various machine learning algorithms, such as logistic regression [10], [42], support vector machines [32] and risk minimization [8], [11], aiming to limit the privacy risk associated with the output model parameters on the training dataset. Our work in this paper is primarily related to [7]. We improve their approach in a number of ways. For example, instead of using traditional (ϵ, δ)-differential privacy, we apply concentrated differential privacy [9], [17] to provide tight cumulative privacy loss estimation over a large number of computations. Furthermore, we characterize the effect of data batching methods on the composition of differential privacy and propose a dynamic privacy budget allocation framework for improving the model accuracy.

VIII. Conclusion

We have presented our approach to differentially private deep learning for model publishing with three original contributions. First, since the training of neural networks involves a large number of iterations, we apply CDP for privacy accounting to achieve tight estimation on privacy loss. Second, we distinguish two different data batching methods and propose privacy accounting methods for each to enable accurate privacy loss estimation. Third, we have implemented several dynamic privacy budget allocation techniques for improving model accuracy over existing uniform budget allocation schemes. Our experiments on multiple datasets demonstrate the effectiveness of dynamic privacy budget allocation.

Acknowledgment

References

[1] https://github.com/tensorflow/models/tree/master/research/differential_privacy.

[2] Breast cancer wisconsin (original) data set. http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29.

[3] Decaying the learning rate. https://www.tensorflow.org/api_guides/python/train.

[4] Imagenet. http://www.image-net.org/.

[5] Input pipeline performance guide. https://www.tensorflow.org/performance/datasets_performance.

[6] Model zoo. http://caffe.berkeleyvision.org/model_zoo.html, accessed 2017.

[7] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep Learning with Differential Privacy. In *Proc. of ACM CCS*, pages 308–318, New York, NY, USA, 2016.

[8] R. Bassily, A. D. Smith, and A. Thakurta. Private empirical risk minimization, revisited. *CoRR*, abs/1405.7085, 2014.

[9] M. Bun and T. Steinke. Concentrated Differential Privacy: Simplifications, Extensions, and Lower Bounds. *CoRR*, 5 2016.

[10] K. Chaudhuri and C. Monteleoni. Privacy-preserving logistic regression. In *Proc of NIPS*, pages 289–296, USA, 2008.

[11] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate. Differentially private empirical risk minimization. *J. Mach. Learn. Res.*, 12:1069–1109, July 2011.

[12] F. Chollet. Xception: Deep learning with depthwise separable convolutions. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1800–1807, 2017.

[13] C. Darken and J. Moody. Note on learning rate schedules for stochastic optimization. In *Proc of Advances in Neural Information Processing Systems (NIPS)*, pages 832–838, San Francisco, CA, USA, 1990.

[14] C. Dwork. Differential privacy. In *Automata, Languages and Programming*, volume 4052 of *Lecture Notes in Computer Science*, pages 1–12. 2006.

[15] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Proc. of Theory of Cryptography*, TCC'06, pages 265–284, Berlin, Heidelberg, 2006.

[16] C. Dwork and A. Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9:211–407, Aug. 2014.

[17] C. Dwork and G. N. Rothblum. Concentrated Differential Privacy. *CoRR*, 3 2016.

[18] C. Dwork, G. N. Rothblum, and S. Vadhan. Boosting and differential privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 51–60, Oct 2010.

[19] M. Fredrikson, S. Jha, and T. Ristenpart. Model Inversion Attacks That Exploit Confidence Information and Basic Countermeasures. In *Proc. of ACM CCS*, pages 1322–1333, New York, NY, USA, 2015.

[20] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *USENIX Security Symposium*, pages 17–32, San Diego, CA, 2014.

[21] A. Ghosh and R. Kleinberg. Inferential privacy guarantees for differentially private mechanisms. *CoRR*, abs/1603.01508, 2016.

[22] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[23] M. Gürbüzbalaban, A. Ozdaglar, and P. Parrilo. Why Random Reshuffling Beats Stochastic Gradient Descent. *ArXiv e-prints*, Oct. 2015.

[24] B. Hitaj, G. Ateniese, and F. Perez-Cruz. Deep models under the gan: Information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 603–618, New York, NY, USA, 2017.

[25] D. Kifer and A. Machanavajjhala. No free lunch in data privacy. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, pages 193–204, New York, NY, USA, 2011.

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

[27] N. Li, W. Qardaji, and D. Su. On sampling, anonymization, and differential privacy or, k-anonymization meets differential privacy. In *Proc. of ASIACCS*, pages 32–33, New York, NY, USA, 2012.

[28] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang. Learning differentially private language models without losing accuracy. *CoRR*, abs/1710.06963, 2017.

[29] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '07, pages 94–103, Washington, DC, USA, 2007.

[30] F. D. McSherry. Privacy integrated queries: An extensible platform for privacy-preserving data analysis. In *Proc. of ACM SIGMOD*, pages 19–30, New York, NY, USA, 2009.

[31] R. M. Rogers, S. P. Vadhan, A. Roth, and J. Ullman. Privacy odometers and filters: Pay-as-you-go composition. In D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, editors, *NIPS*, pages 1921–1929, 2016.

[32] B. I. P. Rubinstein, P. L. Bartlett, L. Huang, and N. Taft. Learning in a large function space: Privacy-preserving mechanisms for svm learning. *CoRR*, abs/0911.5708, 2009.

[33] R. Shokri and V. Shmatikov. Privacy-Preserving Deep Learning. In *Proc. of ACM CCS*, pages 1310–1321, New York, NY, USA, 2015.

[34] R. Shokri, M. Stronati, and V. Shmatikov. Membership Inference Attacks against Machine Learning Models. In *IEEE Symposium on Security and Privacy (S&P)*, 2017.

[35] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[36] C. Song, T. Ristenpart, and V. Shmatikov. Machine learning models that remember too much. In *Proc. of ACM CCS*, pages 587–601, New York, NY, USA, 2017.

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

[38] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart. Stealing Machine Learning Models via Prediction APIs. In *USENIX Security Symposium*, pages 601–618, Austin, TX, 2016.

[39] T. van Erven and P. Harremos. Rényi divergence and kullback-leibler divergence. *IEEE Transactions on Information Theory*, 60(7):3797–3820, July 2014.

[40] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? *CoRR*, abs/1411.1792, 2014.

[41] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. *CoRR*, abs/1611.03530, 2016.

[42] J. Zhang, Z. Zhang, X. Xiao, Y. Yang, and M. Winslett. Functional mechanism: Regression analysis under differential privacy. *Proc. VLDB Endow.*, 5(11):1364–1375, July 2012.

APPENDIX

A. Proof of Theorem 2

Theorem 4. *Suppose that a mechanism $\mathcal{A}$ consists of a sequence of k adaptive mechanism $\mathcal{A}_1, \ldots, \mathcal{A}_k$ where each $\mathcal{A}_i : \prod_{j=1}^{i-1} \mathcal{R}_j \times \mathcal{D} \to \mathcal{R}_i$ and $\mathcal{A}_i$ satisfies ρ_i-zCDP. Let $\mathbb{D}_1, \mathbb{D}_2, \ldots, \mathbb{D}_k$ be the result of a randomized partition of the input domain $\mathbb{D}$. The mechanism $\mathcal{A}(D) = (\mathcal{A}_1(D \cap \mathbb{D}_1), \ldots, \mathcal{A}_k(D \cap \mathbb{D}_k))$ satisfies $\frac{1}{k}(\sum_{j=1}^{k} \rho_i)$-zCDP.*

Proof: Suppose two neighboring datasets D and D'. Without loss of generality, assume that D contains one more element d_e than D'. Let $D_i = D \cap \mathbb{D}_i$ and $D'_i = D' \cap \mathbb{D}_i$. Accordingly, there exists j such that D_j contains one more element than D'_j, and for any $i \neq j$, $D_i = D'_i$. Consider any sequence of outcomes $o = (o_1, \ldots, o_k)$ of $\mathcal{A}_1(D_1), \ldots, \mathcal{A}_k(D_k)$.

Because only D_j is different from D'_j, for any $i \neq j$, we have

$$\Pr[\mathcal{A}_i(D_i) = o_i | \mathcal{A}_{i-1}(D_{i-1}) = o_{i-1}, \ldots, \mathcal{A}_1(D_1) = o_1]$$

equal to

$$\Pr[\mathcal{A}_i(D'_i) = o_i | \mathcal{A}_{i-1}(D'_{i-1}) = o_{i-1}, \ldots, \mathcal{A}_1(D'_1) = o_1]$$

Then, we have

$$L_j^{(o)} \triangleq \log\left(\frac{\Pr(\mathcal{A}(D) = o)}{\Pr(\mathcal{A}(D') = o)}\right)$$

$$= \log\left(\frac{\prod_{i \in [n]} \Pr[\mathcal{A}_i^{D_i} = o_i | \mathcal{A}_{i-1}^{D_{i-1}} = o_{i-1}, .., \mathcal{A}_1^{D_1} = o_1]}{\prod_{i \in [n]} \Pr[\mathcal{A}_i^{D'_i} = o_i | \mathcal{A}_{i-1}^{D'_{i-1}} = o_{i-1}, .., \mathcal{A}_1^{D'_1} = o_1]}\right)$$

$$= \log\left(\frac{\Pr[\mathcal{A}_j^{D_j} = o_j | \mathcal{A}_{j-1}^{D_{j-1}} = o_{j-1}, .., \mathcal{A}_1^{D_1} = o_1]}{\Pr[\mathcal{A}_j^{D'_j} = o_j | \mathcal{A}_{j-1}^{D'_{j-1}} = o_{j-1}, .., \mathcal{A}_1^{D'_1} = o_1]}\right)$$

$$\triangleq c_j(o_j; o_1, \ldots, o_{j-1})$$

where $\mathcal{A}_i^{D_i}$ denotes $\mathcal{A}_i(D_i)$ for short.

Once the prefix $(o_1, \ldots, o_{j-1})$ is fixed,

$$C_j \triangleq c_j(o_j; o_1, \ldots, o_{j-1}) = \log\left(\frac{\Pr(\mathcal{A}_j(D_j) = o_j)}{\Pr(\mathcal{A}_j(D'_j) = o_j)}\right)$$

By the ρ_j-zCDP property of $\mathcal{A}_j$, $\mathbb{E}\left[e^{(\alpha-1)C_j}\right] \leq e^{(\alpha-1)\alpha\rho_j}$, thus

$$\mathbb{E}\left[e^{(\alpha-1)L_j^{(o)}}\right] = \mathbb{E}\left[\exp\left((\alpha-1)C_i\right)\right]$$
$$\leq e^{(\alpha-1)\alpha\rho_j}$$

Because of randomized partition of the input domain $\mathbb{D}$, the extra element d_e of D is randomly mapped to k partitions. Therefore, j is uniformly distributed over $\{1, \ldots, k\}$, and thus the privacy loss $L^{(o)}$ under random data partition is the mixture of independent random variables $L_1^{(o)}, \cdots, L_k^{(o)}$,

$$L^{(o)} = \frac{1}{k}L_1^{(o)} + \ldots + \frac{1}{k}L_k^{(o)}$$

We have

$$\mathbb{E}\left[e^{(\alpha-1)L^{(o)}}\right] = \mathbb{E}\left[\exp\left(\frac{1}{k}(\alpha-1)\sum_{j=1}^{k} L_j^{(o)}\right)\right]$$

$$= \prod_{j=1}^{k} \mathbb{E}\left[\exp\left(\frac{1}{k}(\alpha-1)L_j^{(o)}\right)\right]$$

Let $\beta = \frac{(\alpha-1)}{k} + 1$, and because $L_j^{(o)}$ satisfies zCDP, by (7) then we have

$$= \prod_{j=1}^{k} \mathbb{E}\left[\exp\left((\beta-1)L_j^{(o)}\right)\right]$$

$$\leq \prod_{j=1}^{k} e^{(\beta-1)\beta\rho_j} = e^{(\beta-1)\beta\sum_{j=1}^{k} \rho_j}$$

$$= \exp\left((\alpha-1)\alpha\frac{\sum_{j=1}^{k} \rho_j}{k^2\alpha/(\alpha-1+k)}\right)$$

$$\leq \exp\left((\alpha-1)\alpha\frac{\sum_{j=1}^{k} \rho_j}{k}\right)$$

Therefore, the mechanism $\mathcal{A}(D)$ satisfies $\frac{1}{k}(\sum_{j=1}^{k} \rho_i)$-zCDP. ∎

B. Proof of Lemma 1

Lemma 2. *Suppose that $f : \mathbb{D} \to \mathbb{R}^p$ with $||f(\cdot)||_2 \leq 1$. Consider a mechanism $\mathcal{A}'$ that runs a Gaussian mechanism adding noise $\mathcal{N}(0, \sigma^2\mathbb{I})$ over a random subsample $J \subseteq \mathbb{D}$ where each example is independently sampled with probability q, i.e., $\mathcal{A}'(\mathbb{D}) = \sum_{i \in J} f(x_i) + \mathcal{N}(0, \sigma^2\mathbb{I})$. Let $q \leq \frac{1}{16\sigma}$, then, for two neighboring datasets $\mathbb{D}$ and $\mathbb{D}'$, the α-Rényi divergence between $\mathcal{A}'(\mathbb{D})$ and $\mathcal{A}'(\mathbb{D}')$*

$$D_\alpha(\cdot||\cdot) \leq q^2\alpha/\sigma^2, \quad \text{for} \quad 1 < \alpha \leq \sigma^2 \log\frac{1}{q\sigma} + 1 \quad (18)$$

Proof: By applying the log moment bound $q^2\lambda^2/\sigma^2$ of the privacy loss at the moment order λ given in the proof of Theorem 1 in [7], we have

$$\log \mathbb{E}_{u_0}\left[\left(\frac{qu_1 + (1-q)u_0}{u_0}\right)^\alpha\right] \leq \frac{q^2\alpha(\alpha-1)}{\sigma^2}$$

for $\alpha \leq \sigma^2 \log\frac{1}{q\sigma} + 1$. Then, by definition, $D_\alpha(qu_1 + (1-q)u_0||u_0) \leq q^2\alpha/\sigma^2$. Similarly, we also have $D_\alpha(u_0||qu_1 + (1-q)u_0) \leq q^2\alpha/\sigma^2$. ∎

C. Proof of Theorem 3

Theorem 5. *Let $\widehat{\rho} = q^2/\sigma^2$ in (14). If the mechanism $\mathcal{A}'$ has*

$$D_\alpha(\mathcal{A}'(\mathbb{D})||\mathcal{A}'(\mathbb{D}')) \leq \alpha\widehat{\rho} \quad (19)$$

for $1 < \alpha \leq \sigma^2 \log\frac{1}{q\sigma} + 1$, it satisfies

$$\begin{cases} \left(\widehat{\rho} + 2\sqrt{\widehat{\rho}\log(1/\delta)}, \delta\right) - DP, \text{ if } \delta \geq 1/\exp\left(\widehat{\rho}\sigma^4(\log\frac{1}{q\sigma})^2\right) \\ \left(\widehat{\rho}(\sigma^2 \log\frac{1}{q\sigma} + 1) - \frac{\log\delta}{\sigma^2\log(1/q\sigma)}, \delta\right) - DP, otherwise \end{cases}$$

Proof:

Let $Z = L_{(\mathcal{M}(D)||\mathcal{M}(D'))}^{(o)}$ be privacy loss random variable, then for $1 < \alpha \leq \sigma^2 \log\frac{1}{q\sigma} + 1$,

$$\mathbb{E}\left[e^{(\alpha-1)Z}\right] = e^{(\alpha-1)D_\alpha(M(\mathbb{D})||M(\mathbb{D}'))} \leq e^{(\alpha-1)\alpha\widehat{\rho}} \quad (20)$$

By Markov' inequality,

$$P(Z \geq \epsilon) = P(e^{(\alpha-1)Z} > e^{(\alpha-1)\epsilon}) \leq \frac{\mathbb{E}\left[e^{(\alpha-1)Z}\right]}{e^{(\alpha-1)\epsilon}} \quad (21)$$

Table II: decay rate values for different training times

#epochs	30	40	50	60	70	80	90	100
Time	0.076	0.0441	0.0281	0.019	0.0132	0.0093	0.0067	0.0048
Step	0.5459	0.7008	0.7922	0.851	0.891	0.919	0.94	0.956
Exp	0.0442	0.0282	0.0193	0.0138	0.0101	0.0075	0.0056	0.0041
Poly	6.2077	3.5277	2.1948	1.4317	0.9549	0.6382	0.4167	0.1626

$$\leq \exp((\alpha - 1)(\widehat{\rho}\alpha - \epsilon)) \qquad (22)$$

The unconstrained minimum of function $g(\alpha) = (\alpha - 1)(\widehat{\rho}\alpha - \epsilon)$ occurs at $\alpha^* = (\epsilon + \widehat{\rho})/(2\widehat{\rho})$, and the minimum value is $-(\epsilon - \widehat{\rho})^2/(4\widehat{\rho})$. If $\alpha^* \leq \sigma^2 \log(1/q\sigma) + 1$, this unconstrained minimum corresponds to the constrained minimum as well that is subject to $\alpha \leq \sigma^2 \log(1/q\sigma) + 1$. Let $\delta = \exp(-(\epsilon - \widehat{\rho})^2/(4\widehat{\rho}))$, and then we have $\epsilon = \widehat{\rho} + 2\sqrt{\widehat{\rho}log(1/\delta)}$, which has the same form as in Proposition 1. In this case,

$$\alpha^* = (2\widehat{\rho} + 2\sqrt{\widehat{\rho}log(1/\delta)})/(2\widehat{\rho})$$

, and it requires

$$(2\widehat{\rho} + 2\sqrt{\widehat{\rho}log(1/\delta)})/(2\widehat{\rho}) \leq \sigma^2 \log \frac{1}{q\sigma} + 1 \qquad (23)$$

From (23) we have

$$\delta \geq 1/\exp(\widehat{\rho}\sigma^4(\log(1/q\sigma))^2) \qquad (24)$$

Otherwise, if

$$\delta < 1/\exp((q\sigma \log(1/q\sigma))^2) \qquad (25)$$

which means

$$\alpha^* > \sigma^2 \log(1/q\sigma) + 1$$

Then, because the function $g(\alpha)$ is monotonically decreasing in the interval $(0, \alpha^*]$, the constrained minimum is achieved at the boundary point

$$\alpha^+ = \sigma^2 \log \frac{1}{q\sigma} + 1$$

and accordingly we let $\delta = \exp((\alpha^+ - 1)(\widehat{\rho}\alpha^+ - \epsilon))$ and have

$$\epsilon = \widehat{\rho}\alpha^+ - \frac{log\delta}{\alpha^+ - 1}$$

. ∎

D. ρ-zCDP mechanism $\mathcal{A}$ with random sampling still satisfies ρ-zCDP

Suppose $\mathcal{A}'$ that runs ρ-zCDP mechanism $\mathcal{A}$ on a random subsample of dataset $\mathbb{D}$. Following the result (11) in the paper, the proof is as follows:

Because $\Pr(\Lambda(\mathbb{D}') = T'|d_e \in T') = \Pr(\Lambda(\mathbb{D}) = T)$ and $\mathcal{A}$ satisfies ρ-zCDP, Rényi divergence

$$D_\alpha(u_0||u_1) \leq D_\alpha(\mathcal{A}(T)||\mathcal{A}(T')) \leq \alpha\rho \qquad (26)$$

$$D_\alpha(u_1||u_0) \leq D_\alpha(\mathcal{A}(T')||\mathcal{A}(T)) \leq \alpha\rho \qquad (27)$$

Using jointly quasi-convexity of Rényi divergence [39], we have

$$D_\alpha(u_0||qu_1 + (1 - q)u_0) \leq D_\alpha(u_0||u_1) \leq \alpha\rho \qquad (28)$$

$$D_\alpha(qu_1 + (1 - q)u_0 \ || \ u_0) \leq D_\alpha(u_1||u_0) \leq \alpha\rho \qquad (29)$$

and thus $\mathcal{A}'$ still satisfies ρ-zCDP.

E. The Impact of number of hidden layers for MNIST

See Table III.

Table III: Accuracy results under different number of layers

Layer numbers	1	2	3
training accuracy	0.9315	0.9228	0.902
testing accuracy	0.931	0.921	0.898

F. Decay rate values for different training times

Table II list the values of decay rate used for each decay function to achieve the corresponding training times.

G. Schedule parameters for the Cancer and CIFAR-10 datasets

See Table IV.

Table IV: Schedule parameters

Dataset	Uniform	Exp Decay	Validation
Cancer	σ=25, #epochs=500	σ_0=30, k=0.001	σ_0=35,k=0.99, period=50,δ =0.01, m=1
CIFAR-10	σ=8, #epochs=200	σ_0=6, k=0.001	σ_0=6,k=0.99, period=10,δ =0.05, m=5