Federated f-Differential Privacy

Qinqing Zheng **Shuxiao Chen** **Qi Long** **Weijie J. Su**
University of Pennsylvania

Abstract

Federated learning (FL) is a training paradigm where the clients collaboratively learn models by repeatedly sharing information without compromising much on the privacy of their local sensitive data. In this paper, we introduce *federated f-differential privacy*, a new notion specifically tailored to the federated setting, based on the framework of Gaussian differential privacy. Federated f-differential privacy operates on *record level*: it provides the privacy guarantee on each individual record of one client's data against adversaries. We then propose a generic private federated learning framework `PriFedSync` that accommodates a large family of state-of-the-art FL algorithms, which provably achieves federated f-differential privacy. Finally, we empirically demonstrate the trade-off between privacy guarantee and prediction performance for models trained by `PriFedSync` in computer vision tasks.

1 Introduction

Federated learning (McMahan et al., 2017) is an emerging paradigm that enables multiple clients to collaboratively learn prediction models without explicitly sharing data. Unlike traditional distributed training approaches that upload all the data to central servers, federated learning performs on-device training and only some summaries of local data or local models are exchanged among clients. Typically, the clients upload their local models to the server and share the global averaging in a repeated manner. This offers plausible solutions to address the critical data privacy issue: sensitive information about individuals such as typing history, shopping transactions, geographical locations, medical records, would stay localized.

Nonetheless, a malicious client who participates in the federated learning might still be able to learn information about the other clients' data through the shared model's weights. This is because it is possible for an adversary to learn about or even identify certain individuals by simply tweaking the input datasets and probing the output of the algorithm (Fredrikson et al., 2015; Shokri et al., 2017). This gives rise to a pressing call for privacy-preserving federated learning algorithms. Accordingly, we urgently need a rigorous and principled framework to enhance data privacy, and to quantitatively answer the important questions:

Can another client identify the presence or absence of any individual record in my data in federated learning? Worse, what if all the other clients ally each other to attack my data?

A number of works have tried to answer *similar* questions from different perspectives (McMahan et al., 2018; Geyer et al., 2017; Li et al., 2019; Singh et al., 2020), and numerous privacy notions and associated approaches are proposed to address *related* problems, yet *none* of them have answered these questions fully and directly. To the best of our knowledge, all existing works plainly generalize the classical differential privacy definition to the federated setting: an adversary can remove one client's whole dataset, and this type of attack would not incur massive changes to the output of the algorithm. The resulting privacy guarantee executes at the user level: whether a client has participated in the training can not be inferred by adversaries, and the client's whole dataset is private.

While the user level privacy has important applications in federated learning, it is complementary and equally important to consider weaker privacy notions at the record level. First, privacy is generally at odds with performance. A user level privacy

Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130.

guarantee is usually too strong and one often seeks a weaker notion that protects privacy from more practical attacks (Li et al., 2019). More importantly, consider the case whee multiple hospitals in different countries would like to collaboratively learn prediction models for COVID-19. In this example, whether a hospital participates in this collaboration is not a sensitive information at all, and what really needs to be protected is the privacy of each patient. This is a regime that a record level privacy notion shines.

In this work, we introduce a fine-grained privacy notion, called *weak federated f-differential privacy*, that protects *each individual record* of one client's data. We work on the attack model that an adversary can manipulate one single record of the client's dataset and provide privacy guarantees for this case. We propose a unified private federated learning framework `PriFedSync` where a large family of federated learning algorithms kick in. Besides, we give an extended privacy notion, called *strong federated f-differential privacy*, to address the case where multiple malicious clients jointly attack a client, which has not been considered in any previous work. Our major contributions are as follows.

1. We introduce two privacy notions, *weak federated f-differential privacy* and *strong federated f-differential privacy*, that describe the privacy guarantee against an individual adversary and against a group of adversaries, respectively. Both notions are of the finest resolution in the sense that they protect individual records of one client's data. The privacy definition that we rely on is f-differential privacy, in particular its sub-family of *Gaussian differential privacy* (GDP) (Dong et al., 2019).

2. We propose a generic federated learning framework `PriFedSync` that contains the state-of-the-art federated learning algorithms. The framework does not assume a trusted central aggregator. It can accommodate both personalized or non-personalized approaches. We exploit the composition theorem of GDP to analyze the privacy guarantee of `PriFedSync` and prove its asymptotic convergence.

3. We conduct numerical experiments to illustrate our privacy notions and compare the performance of private models with non-private counterparts. When the data is heterogeneous across clients, our personalized approach demonstrates significant improvement over the global model. We also demonstrate the trade-offs between privacy and accuracy, and privacy and computation, through our experiments.

The rest of this paper is organized as follows. We give a brief review of the research on federated learning and differential privacy in Section 1.1. Section 2 introduces our training framework. Section 3 presents the privacy notion and analysis. Section 4 presents the numerical experiments.

1.1 Related Work

There is a growing body of work that have looked at privacy properties in the context of federated learning. McMahan et al. (2018) introduces two algorithms, *differentially private federated stochastic gradient descent* (`DP-FedSGD`) and *differentially private federated averaging* (`DP-FedAvg`), and studies their privacy properties. The privacy notion is defined on user level. Namely, two datasets S and S' are said to be neighboring if S' can be obtained by completely removing one client's data from S. Such an attack might be impractical for real world applications. Algorithmically, `DP-FedSGD` is a direct extension of "non-federated" `DP-SGD` (Abadi et al., 2016) to the distributed optimization setting, where the gradients of each client is clipped and aggregated in every iteration, whereas `DP-FedAvg` performs approximated `DP-SGD` on the server. In essence, the differences of local models before and after local training are treated as the surrogates of gradients and sent to the server. A similar algorithm approximating `DP-SGD` is proposed in Geyer et al. (2017). Singh et al. (2020) uses an algorithm similar to `DP-FedSGD` for the architecture search problem, and their privacy guarantee acts on user level too. Li et al. (2019) studies the online transfer learning and introduces a notion called task global privacy that works on record level. However, the online setting assumes the client only interacts with the server once and does not extend to the federated setting. Truex et al. (2019) generalizes the central differential privacy into the distributed setting, which can be considered as training a single shared private model when the dataset is split into several partitions on different machines. Even though the authors consider privacy at record level, this work does not have setups such as individual clients have their own models, client sampling, etc.

One privacy notion that is related to the general concept of privacy in federated learning is local differential privacy (Evfimievski et al., 2003; Kasiviswanathan et al., 2011). Local differential privacy does not assume a trusted data aggregator. Each data record is randomly perturbed before sending to the data aggregator, and the aggregator build models using the noisy data. The perturbation algorithm is locally differentially private if the out-

puts of any pair of possible data records are indistinguishable. Although conceptual connected, local differential privacy does not perfectly extend to the general federated learning environment. Under the local differential privacy framework, the noisy data are finally centralized in a central aggregator, where all the training happens; whereas a general form of federated learning allows the participants have their own control of data and models. Besides, local differential privacy is a strong notion that often requires a large amount of noise and thus leads to degraded model performance.

Another related notion is joint differential privacy, proposed by Kearns et al. (2014) to study the behavior of "recommender mechanisms" for large games. It has been applied to the context of private convex programming for problems whose solution can be divided between different agents (Hsu et al., 2016a,b), e.g. the multi-commodity flow problem. Informally, joint differential privacy ensures the joint distribution of the outputs for Agent $j \neq i$ to be insensitive to the input provided by Agent i. It is similar to the our one-vs.-all notion *strong federated f-differential privacy* (see Definition 3.6), but acts on user level.

Despite the granularity and concrete notion of the privacy guarantee, a formal privacy definition is needed to precisely quantify the privacy loss. The most popular statistical privacy definition to date is (ϵ, δ)-differential privacy (Dwork et al., 2006a,b). It is widely applied in industrial applications and academic research, including some previous work on private federated learning (McMahan et al., 2017; Geyer et al., 2017; Li et al., 2019). Unfortunately, this privacy definition does not well handle the cumulative privacy loss under the composition of private algorithms (Kairouz et al., 2017; Murtagh and Vadhan, 2016), which is a fundamental problem to address in privacy analysis, and also needed in analyzing the federated learning algorithms. The need for a better treatment of composition has motivated much work in proposing divergence-based relaxations of (ϵ, δ)-differential privacy relaxations (Dwork and Rothblum, 2016; Bun and Steinke, 2016; Mironov, 2017; Bun et al., 2018). Meanwhile, another line of research has established the connection between differential privacy and hypothesis testing (Wasserman and Zhou, 2010; Kairouz et al., 2017; Liu et al., 2019; Balle et al., 2019). Recently, Dong et al. (2019) proposes a hypothesis testing-based privacy notion termed *f-differential privacy*. This privacy definition characterizes the privacy guarantee using the trade-off between type I and type II errors given via the associated hypothesis testing problem. In the case of testing for normal distributions, f-differential privacy reduces to Gaussian differential privacy. Owing to its lossless reasoning about composition and privacy amplification by subsampling, the use of f-differential privacy gives sharp, analytically tractable expressions for the privacy guarantees of training deep learning models (Bu et al., 2020) (see also (Zheng et al., 2020)). Throughout this paper, we use GDP as our privacy analysis framework.

2 Private Federated Learning

Let m denote the number of clients. Each Client i has access to its local dataset $S^{(i)}$, where the data are i.i.d sampled from local distribution $\mathcal{D}_i$. The classic federated learning algorithms (McMahan et al., 2017; Konečnỳ et al., 2016) aim at learning one global model $\tilde{w}^{\text{global}}$ that performs well over all the clients. This implicitly makes an underlying assumption that the data are homogeneous, i.e., $\mathcal{D}_1 = \cdots = \mathcal{D}_m$, yet in practice data might not be identically distributed across clients. To take into account of the heterogeneity of user data distributions, there is a surge of interest to assume non-identical data distributions with the possibility of $\mathcal{D}_i \neq \mathcal{D}_j$, and learn personalized models (Dinh et al., 2020; Huang et al., 2020; Hanzely and Richtárik, 2020; Deng et al., 2020).

We propose a unified framework `PriFedSync` that addresses both heterogeneous and homogeneous settings, see Algorithm 1. Each Client i will obtain a specific model $\tilde{w}^{(i)}$, and a global model $\tilde{w}^{\text{global}}$ is still formed and utilized. The homogeneous setting boils down to a special case where $\tilde{w}^{(i)} = \tilde{w}^{\text{global}}, i \in [m]$ (we use $[m]$ to denote $\{1, \ldots, m\}$). `PriFedSync` subsumes a large family of existing federated learning algorithms, including `FedAvg` (McMahan et al., 2017) and many others (Li et al., 2018; Dinh et al., 2020; Huang et al., 2020; Hanzely and Richtárik, 2020; Deng et al., 2020).

In `PriFedSync`, all the clients start from the same model w_0. To mimic the practical behavior that not all the clients sync with the server simultaneously, in every synchronization round we sample a subset of clients to perform local training and sync with the server. If Client i is selected, it pulls a helper model $\tilde{h}^{(i)}$ from the server and then performs local private training for K iterations. The helper model $\tilde{h}^{(i)}$ can be the global aggregation $\tilde{w}^{\text{global}}$ (McMahan et al., 2017; Li et al., 2018), or personalized (Dinh et al., 2020; Huang et al., 2020). Various ways have been proposed to utilize $\tilde{h}^{(i)}$ to improve local training, including initializing local models (Dinh et al., 2020; Hanzely and Richtárik, 2020), regularizing lo-

Algorithm 1: `PriFedSync` Framework

Input: *initialization* w_0, *number of local iterations* K, *number of synchronization rounds* R, *sync probability* p

Initialization: $w^{(i)}, \tilde{h}^{(i)} \leftarrow w_0$

for $r = 1, \dots R$ **do**

  Poisson sample a subset of clients $\Omega \subseteq \{1, \dots, m\}$ with probability p

  Client $i \in \Omega$ **do in parallel**

    `// local training for` K `iterations`

    $\tilde{w}^{(i)} \leftarrow$ LocalPrivateTraining$(S^{(i)}, \tilde{h}^{(i)}, K)$

    Send $\tilde{w}^{(i)}$ to Server

  Server do

    $\tilde{w}^{\text{global}} \leftarrow (1-\eta)\tilde{w}^{\text{global}} + \eta \frac{1}{|\Omega|} \sum_{i \in \Omega} \tilde{w}^{(i)}$

    $\tilde{h}^{(i)} \leftarrow F_i(\tilde{w}^{\text{global}}),\ i \in \Omega$

    Push noisy helper model $\tilde{h}^{(i)}$ to Client i, $i \in \Omega$

cal training (Li et al., 2018; Huang et al., 2020), and iterative interpolating with local updates (Deng et al., 2020).

The local private training can be carried out in different ways too. For instance, one can use noiseless local training and perturb the model before synchronization using Laplacian or Gaussian mechanism. Alternatively, one can conduct `DP-SGD` (Abadi et al., 2016) directly, where the gradient is perturbed in each iteration. The disadvantage of `DP-SGD` is that it is slow in computation, due to its need of clipping the per-sample gradient at every iteration. However, we observed that `DP-SGD` usually leads to better prediction accuracy, therefore we shall use `DP-SGD` for our analysis and experiments.

Next, Client i sends the private model $\tilde{w}^{(i)}$ to the server. The server aggregates the received models and then updates the corresponding helper models. There are plenty of ways of computing the helper model. If the F_i function is the identity map $F_i(w) = w, i \in [m]$, all the clients will receive the same helper model $\tilde{w}^{\text{global}}$. This is the setup used in `FedAvg` (McMahan et al., 2017). Another simple but effective observation is that $\tilde{h}^{(i)}$ is a convex combination of the noisy global model w^{global} and local model $w^{(i)}$ (Dinh et al., 2020; Hanzely and Richtárik, 2020):

$$F_i(\tilde{w}^{\text{global}}) = (1-\alpha_i)w^{(i)} + \alpha_i \tilde{w}^{\text{global}}. \tag{2.1}$$

There are more sophisticated constructions of personalized helper models that fit in our framework, for example, an attention-based weighted averaging (Huang et al., 2020).

We close this section with an overview discussion of the privacy guarantee of `PriFedSync`.

1. It is easy to see that Client i can only probe the dataset of Client j through the helper model $\tilde{h}^{(i)}$. This becomes the focal point for our analysis throughout this paper.

2. Given a global model $\tilde{w}^{\text{global}}$, the helper model $\tilde{h}^{(i)}$ is a transformation of $\tilde{w}^{\text{global}}$ through the mapping F_i. Regardless of the form of F_i, this step would not cause additional privacy leakage since differential privacy is immune to post-processing (Dwork and Roth, 2014). It is then natural to ask the following questions:

(i) Why not just compute a noiseless global model w^{global} and inject noise before or after the transformation? For instance, on the server one can conduct

$$w^{\text{global}} \leftarrow (1-\eta)w^{\text{global}} + \eta \frac{1}{|\Omega|} \sum_{i \in \Omega} w^{(i)},$$
$$\tilde{h}^{(i)} \leftarrow F_i\big(w^{\text{global}} + \mathcal{N}(0, \sigma^2 I)\big), i \in \Omega.$$

(ii) Is it equivalent to directly send $\tilde{w}^{\text{global}}$ to the clients and let them apply the transformation F_i's themselves?

The procedure in (i) indeed protects the privacy of Client i and reduces the computational burden incurred by `DP-SGD`. However, computing a noiseless global model will require all the clients send noiseless local models to the server, which imposes an extra assumption about a trustworthy server. The answer to (ii) depends on the concrete form of F_i. If the mapping F_i is free of other private local models, deterministic, and invertible, the privacy cost before and after applying F_i is the same. In this scenario, there is no difference between sending $\tilde{w}^{\text{global}}$ or $\tilde{h}^{(i)}$. Nevertheless, post-processing might be able to amplify the privacy. Consider a constant function F_i that outputs the zero vector for any input. This simple function achieves perfect privacy. For those cases, sending $\tilde{w}^{\text{global}}$ will be less private then sending $\tilde{h}^{(i)}$. To keep our analysis general for all algorithms that fit in `PriFedSync`, we shall assume no knowledge of F_i in our analysis. For a specific algorithm, potential tighter bounds might be obtained by taking prior knowledge of F_i.

3 Privacy Analysis

We first review Gaussian differential privacy in Section 3.1, which is the analysis tool we exploit. Next,

we introduce our private notations in Section 3.2 and analyze the privacy guarantee of `PriFedSync` in Section 3.3.

3.1 Preliminaries

Let us start from the hypothesis testing interpretation of differential privacy, which is the foundation of GDP. Let $\mathcal{A}$ denote a randomized algorithm that takes a dataset S as input. S' is a neighboring dataset of S in the sense that S and S' differ in only one individual. Let P and Q denote the probability distribution of $\mathcal{A}(S)$ and $\mathcal{A}(S')$, respectively. Differential privacy attempts to measure the difficulty for an adversary to identify the presence or absence of any individual in S via leveraging the output of $\mathcal{A}$. Equivalently, an adversary performs the following hypothesis testing problem (Wasserman and Zhou, 2010):

$$H_0 : \text{output} \sim P \quad \text{vs} \quad H_1 : \text{output} \sim Q.$$

Intuitively, a privacy breach occurs if the adversary makes the right decision, and the privacy guarantee of $\mathcal{A}$ boils down to the difficulty for an adversary to tell the two distributions apart. Dong et al. (2019) proposes to use the trade-off between type I and type II errors of the optimal likelihood ratio tests at level α as a measure of the privacy guarantee, where α ranges from 0 to 1. Formally, let ϕ be a rejection rule for testing against H_0 against H_1. The type I and type II error of ϕ are $\mathbb{E}_P(\phi)$ and $1 - \mathbb{E}_Q(\phi)$, respectively. The trade-off function $T : [0, 1] \to [0, 1]$ between the two probability distributions P and Q is defined as

$$T(P, Q)(\alpha) = \inf_{\phi} \{1 - \mathbb{E}_Q(\phi) : \mathbb{E}_P(\phi) \leq \alpha\}.$$

In short, for a fixed significance level α, $T(P, Q)(\alpha)$ is the minimum type II error that a test can achieve at that level. The optimal tests are given by the Neyman–Pearson lemma, and can be interpreted as the most powerful adversaries. Let us define the relation $f \geq g$ if $f(\alpha) \geq g(\alpha)$ for all $0 \leq \alpha \leq 1$. Intuitively speaking, a larger trade-off function implies the more private the associated algorithm is. A special case of interest is when the two distributions are the same and perfect privacy is attained. The corresponding trade-off function is $T(P, P)(\alpha) = 1 - \alpha$, which we denote by $\mathrm{Id}(\alpha)$. With the above definitions in place, Dong et al. (2019) introduces the following privacy definition, with a little abuse of notation by using $\mathcal{A}(S)$ to denote the output distribution of algorithm $\mathcal{A}$ on input dataset S.

Definition 3.1. Let f be a trade-off function. An algorithm $\mathcal{A}$ is f-differentially private if $T(\mathcal{A}(S), \mathcal{A}(S')) \geq f$ for any pair of neighboring datasets S and S'.

When the trade-off function is defined between two Gaussian distributions, we obtain a subfamily of f-differential privacy guarantees called Gaussian differential privacy.

Definition 3.2. Let Φ denote the cumulative distribution function of the standard normal distribution. For $\mu \geq 0$, let $G_\mu := T(\mathcal{N}(0, 1), \mathcal{N}(\mu, 1)) \equiv \Phi(\Phi^{-1}(1 - \alpha) - \mu)$. An algorithm $\mathcal{A}$ is μ-GDP if $T(\mathcal{A}(S), \mathcal{A}(S')) \geq G_\mu$ for any pair of neighboring datasets S and S'.

One advantage of f-differential privacy is that the composition of algorithms can be neatly handled. The composition primitive refers to an algorithm $\mathcal{A}$ that consists of R algorithms $\mathcal{A}_1, \ldots, \mathcal{A}_R$, where $\mathcal{A}_i$ observes both the input dataset and output from all previous algorithms. Let $f_1 = T(P_1, Q_1)$ and $f_2 = T(P_2, Q_2)$, Dong et al. (2019) defines a binary operator $\otimes$ on trade-off functions such that $f_1 \otimes f_2 = T(P_1 \times P_2, Q_1 \times Q_2)$, where $P_1 \times P_2$ is the distribution product. This operator is commutative and associative, and provides elegant formulations for the composition of private algorithms.

Lemma 3.3 (Dong et al. (2019))**.** *If $\mathcal{A}_i$ is f_i-differentially private for $1 \leq i \leq R$, then the composed algorithm $\mathcal{A}$ is $f_1 \otimes \cdots \otimes f_R$-differentially private.*

Lemma 3.4 (Dong et al. (2019))**.** *The R-fold composition of μ_i-GDP algorithms is $\sqrt{\mu_1^2 + \cdots + \mu_n^2}$-GDP.*

3.2 Federated f-Differential Privacy

Section 2 has discussed that the privacy leakage of Client j to Client i is determined by the helper model $\tilde{h}^{(i)}$, which motivates the following definitions.

Recall that $S^{(j)}$ is the dataset of Client j. Let $S^{'(j)}$ denote a neighboring dataset of $S^{(j)}$, i.e., $S^{'(j)}$ and $S^{(j)}$ differ by only one entry. Let $\boldsymbol{S} = \left(S^{(1)}, \ldots, S^{(m)}\right)$ denote the joint dataset across clients. Let $M(\cdot) = (M_1(\cdot), \ldots, M_m(\cdot))$ be the randomized federated algorithm that returns the helper models to clients: $M_i(\boldsymbol{S}) = \tilde{h}^{(i)}$ is the helper model for Client i. Note that for M_i, the usage of $S^{(j)}_{j \neq i}$ is implicit: Client i is blind to those datasets. We write $\boldsymbol{S}^{'j}$ if it is neighboring with $\boldsymbol{S}$ in the j-th component: $\boldsymbol{S}^{'j} = (S^{(1)}, \ldots, S^{'(j)}, \ldots, S^{(n)})$. The following two definitions quantitatively describe how well every client could protect her/his own data against the other clients.

Definition 3.5. A randomized federated learning algorithm M satisfies the *weak federated* f-

differential privacy if for any $i \neq j$, it holds that $T\big(M_i(\boldsymbol{S}), M_i(\boldsymbol{S}'^j)\big) \geq f$.

Definition 3.6. Let M_{-j} denote the randomized output of all the helper models except j. M satisfies the *strong federated f-differential privacy* if it holds that for any j, $T\big(M_{-j}(\boldsymbol{S}), M_{-j}(\boldsymbol{S}'^j)\big) \geq f$. This is equivalent to $T\big(\prod_{i\neq j} M_i(\boldsymbol{S}), \prod_{i\neq j} M_i(\boldsymbol{S}'^j)\big) \geq f$.

We remark that Definition 3.5 is a one-vs.-one privacy notion. Under this notation, every client is protected from the attack from any other malicious client. Definition 3.6 is a one-vs.-all privacy notion. In the worst case, the other clients would make allies and attack Client i together. An algorithm M satisfying Definition 3.6 could guarantee the privacy of Client i even in this situation. In other words, if M satisfies the strong federated f-differential privacy, then it satisfies the weak federated f-differential privacy.

3.3 Analysis

Let $\tilde{H}_i$ denote the update of $\tilde{h}^{(i)}$ on the server. In practice, if Client i is not sampled for synchronization, the algorithm does not release a model to Client i, thus the perfect privacy of all the other clients' data is achieved. In the privacy analysis, this is equivalent to releasing a constant number that carries zero information. Letting $\boldsymbol{S}_\Omega$ denote the subsampled dataset, we can write this update as:

$$\tilde{H}_i(\boldsymbol{S}_\Omega) = \begin{cases} \tilde{h}^{(i)}(\boldsymbol{S}_\Omega), & \text{if } i \in \Omega, \\ \mathbf{0}, & \text{otherwise.} \end{cases} \tag{3.1}$$

Let $\texttt{Sample}_p$ denote the Possion subsampling of clients for synchronization. The update of $\tilde{h}^{(i)}$ for one synchronization round is the subsampled algorithm $\tilde{H}_i \circ \texttt{Sample}_p$. We remark that the subsampling step $\texttt{Sample}_p(S) = \Omega$ is an intermediate step that is not released, and the subsampled algorithm $\tilde{H}_i \circ \texttt{Sample}_p$ should be considered as a whole. Our target to analyze is essentially the composition of R copies of $\tilde{H}_i \circ \texttt{Sample}_p$:

$$\begin{aligned} &T\big(M_i(\boldsymbol{S}), M_i(\boldsymbol{S}'^j)\big) \\ =&T\big((\tilde{H}_i \circ \texttt{Sample}_p)^{\otimes R}(\boldsymbol{S}), (\tilde{H}_i \circ \texttt{Sample}_p)^{\otimes R}(\boldsymbol{S}'^j)\big). \end{aligned} \tag{3.2}$$

The analysis has three steps. We first need to understand the privacy guarantee of the algorithm $\tilde{H}_i$, without sampling. The second step is to figure out the guarantee of the subsampled algorithm $\tilde{H}_i \circ \texttt{Sample}_p$. Last, we apply the composition theorem of f-differential privacy to obtain the final guarantee. The results are presented in Lemma 3.7, Lemma 3.8, and Theorem 1 in order.

Lemma 3.7. *For any Client j, suppose the local training of $\tilde{w}^{(j)}$ is f_j-differentially private. It holds that*

$$T\big(\tilde{H}_i(\boldsymbol{S}), \tilde{H}_i(\boldsymbol{S}'^j)\big) \geq f_j,\ i \in [m].$$

Proof. See Appendix A. □

This lemma implies that for any Client j, the privacy guarantee holds uniformly the same for all the other clients. Intuitively, the privacy loss is determined once Client j dispatches $\tilde{w}^{(j)}$, and the subsequent post-processing of $\tilde{w}^{(j)}$ will incur no extra privacy loss. The privacy leakage to the other clients will only differ if Client j sends different models with different levels of noise to the other clients, explicitly or implicitly. Since each client only communicates with the server in `PriFedSync`, we can guarantee the privacy protection is uniform over all the other clients.

Next, we analyze the subsampled algorithm $\tilde{H}_i \circ \texttt{Sample}_p$. Compared with the original algorithm, subsampling amplifies the privacy guarantee. Such amplification is due to the fact that if Client j is not included in one round of synchronization, it enjoys perfect privacy for that round. Our results are described formally in the following lemma.

Lemma 3.8. *Let $g_{p,j} = \max(f_j, 1-\alpha-p^2)$. Suppose the local training algorithm of $\tilde{w}^{(j)}$ is f_j-differentially private. Consider the subsampled algorithm $\tilde{H}_i \circ \texttt{Sample}_p$ with $0 \leq p \leq 1$. For any $i \in [m]$, it holds that*

$$T\big(\tilde{H}_i \circ \texttt{Sample}_p(\boldsymbol{S}), \tilde{H}_i \circ \texttt{Sample}_p(\boldsymbol{S}'^j)\big) \geq g_{p,j}.$$

Proof. See Appendix B. □

We emphasize that the technical needs for analyzing the client sampling of `PriFedSync` is different from the analysis of private SGD with Poisson sampling (Bu et al., 2020), and the existing results do not directly apply to our case. The main difference is that for `PriFedSync`, the privacy loss of Client j to Client i is affected by whether i and j are both sampled in Ω. From the hypothesis testing point of view, the two distributions the adversary is trying to tell apart, $\tilde{H}_i \circ \texttt{Sample}_p(\boldsymbol{S})$ and $\tilde{H}_i \circ \texttt{Sample}_p(\boldsymbol{S}'^j)$, are both mixture models of two groups: one group contains the cases both i and j are sampled, the other group contains the other cases. Whereas, for analyzing private SGD, only one of the two distributions need to be divided into two groups.

Finally, we apply the composition theorem of f-differential privacy (Lemma 3.3) to obtain the following results.

Algorithm 2: Example Local Training of Client j using NoisySGD

Input: *loss* L, *dataset* $S^{(j)}$, *helper model* $\tilde{h}^{(j)}$, *batch size* B_j, *noise scale* σ_j, *maximum gradient norm* C, *learning rates* $\gamma_1, \ldots, \gamma_K$

Initialize: $w^{(j)} \leftarrow \tilde{h}^{(j)}$

for $k = 1, \ldots K$ **do**

  Sample $I \subseteq \{1, \ldots, |S^{(j)}|\}$ with size B_j uniformly at random

  `// Compute and clip the per-sample gradient`

  for $\ell \in I$ **do**

    $v_\ell = \nabla L(w^{(j)}, x_\ell, y_\ell)$

    $v_\ell \leftarrow v_\ell / \max\{1, \|v_\ell\| / C\}$

  $w^{(j)} \leftarrow w^{(j)} - \frac{\gamma_k}{B}\left(\sum_{\ell \in I} v_\ell + \mathcal{N}(0, 4C^2\sigma_j^2 I)\right)$

Theorem 1. *Let* $g_{p,j} = \max(f_j, 1 - \alpha - p^2)$ *be defined as in Lemma 3.8. It holds that*

$$T\big(M_i(\boldsymbol{S}), M_i(\boldsymbol{S}'^j)\big) \geq g_{p,j}^{\otimes R}, \ i \in [m].$$

Consequently, Algorithm 1 satisfies weak federated f*-differential privacy for* $f = g_{p,j_{\min}}^{\otimes R}$, *where* $g_{p,j_{\min}} = \min\{g_{p,1}, \ldots, g_{p,m}\}$. *It also satisfies strong federated* $g_{p,j_{\min}}^{\otimes (m-1)R}$*-differential privacy.*

Proof. See Appendix C. □

3.4 Local Private Training

In this section, we present an example local training algorithm using noisy SGD as the optimizer, see Algorithm 2. We analyze its privacy guarantee and present a final privacy bound of `PriFedSync` after injecting it into Algorithm 1. Although Algorithm 2 uses SGD as the optimizer, our results hold for a large number of other optimizers, including Adam (Kingma and Ba, 2014), AdaGrad (Duchi et al., 2011), Momentum SGD (Qian, 1999), etc. In brief, this is because the statistics like the momentum, the running mean of the gradient, are deterministic functions of the noisy gradient, thus no additional privacy loss would be incurred for those computations.

Let $f_p = pf + (1-p)\mathrm{Id}$ for some $p \in [0,1]$. Let f_p^{-1} be the inverse function of f_p: $f_p^{-1}(\alpha) = \inf_{t \in [0,1]}\{f_p(t) \leq \alpha\}$. Define a trade-off function $C_p(f) = \min\{f_p, f_p^{-1}\}^{**}$ where f^{**} denotes the double conjugate of f. The function f_p is asymmetric in general but $C_p(f)$ is symmetric, see Figure E.1.

Theorem 2. *Suppose Algorithm 2 is used in Algorithm 1 for the local private training. It holds that for any Client* j,

$$T\big(M_i(\boldsymbol{S}), M_i(\boldsymbol{S}'^j)\big) \geq C_{B_j/n_j}(G_{1/\sigma_j})^{\otimes KR}, \ i \in [m].$$

Furthermore, if $\frac{B_j}{n_j}\sqrt{KR} \to c_j$ *as* $\sqrt{KR} \to \infty$, *then* $C_{B_j/n_j}(G_{1/\sigma_j})^{\otimes KR} \to G_{\mu_j}$ *where*

$$\mu_j = \sqrt{2}c_j\sqrt{e^{\sigma_j^{-2}}\Phi(1.5\sigma_j^{-1}) + 3\Phi(-0.5\sigma_j^{-1}) - 2}.$$

Consequently, Algorithm 1 satisfies weak federated $G_{\mu_{\max}}$*-differential privacy and strong federated* $G_{\sqrt{m-1}\mu_{\max}}$*-differential privacy, where* $\mu_{\max} = \max\{\mu_1, \ldots, \mu_m\}$.

Proof. See Appendix D. □

4 Experiments

We use Algorithms 1 and 2 to train private deep learning models for two computer vision tasks: MNIST digit recoginition (LeCun, 1998) and CIFAR-10 object classification (Krizhevsky and Hinton, 2009)[1]. To simulate the heterogeneous data distributions, we make non-IID partitions of the datasets, see below for the detailed descriptions. For all the experiments, we fix the aggregation parameter $\eta = 1$, use the interpolation method as in Equation (2.1) with $\alpha = 0.1$ to compute the helper models, and clip the gradient with maximum norm $C = 1$ when training private models. For both tasks, we report the average testing accuracy along with the privacy guarantees we obtained, and compare with the non-private results under the same setting. The algorithms and models we use might not yield the best possible prediction accuracy, but they are sufficient for the purposes of illustrating our private notion and investigating the relative performance for private and non-private algorithms.

4.1 Non-IID MNIST

The MNIST dataset contains 60,000 training images and 10,000 testing images. We use a setup similar to (McMahan et al., 2017) to partition the data for 100 clients. We sort the training data by digit label and evenly divide it into 400 shards. Each of 100 clients is assigned four random shards of the data, so that most of the clients have examples of three or four digits. For testing, each client will sample 200 examples with the same label she/he has seen in training[2]. We use a CNN model with two convo-

[1] Our code is available at `https://github.com/enosair/federated-fdp`.

[2] In contrast to some previous works where only the training data is non-IID, our testing data is also not identically distributed across the clients.

p	σ	R	$\mu_{\max}$	Test Acc	Non-Pri Acc
1.0	1.0	93	2.71	90.03	98.74
	0.9	83	3.10	90.25	98.72
	0.75	64	3.96	90.10	98.55
0.5	1.0	194	3.92	90.02	98.90
	0.9	176	4.51	90.02	98.83
	0.75	127	5.58	90.11	98.54
0.25	1.0	386	5.52	90.00	98.75
	0.9	325	6.13	90.00	98.75
	0.75	245	7.75	90.04	98.55

Table 1: MNIST experiment results: the round of synchronization and corresponding privacy parameter when the average test accuracy reaches 90%. The privacy parameter is computed as in Theorem 2.

σ	B	K	R	Total Iter.	Total Ex.	$\mu_{\max}$
1	8	76	266	20216	161728	3.24
	16	38	194	7372	117952	3.92
0.9	8	76	229	17404	139232	3.64
	16	38	176	6688	107008	4.51
0.75	8	76	191	14516	116128	4.84
	16	38	127	4826	77216	5.58

Table 2: The trade-off between privacy and computation. The per-client total number of iterations is KR, the per-client total number of examples is BKR, and the results are reported when the average test accuracy reaches 90%. The privacy parameter $\mu_{\max}$ for small batch size ($B = 8$) runs is roughly $0.83\times$ as large as obtained by the $B = 16$ runs, yet the $B = 8$ runs process approximately $1.39\times$ total number of data samples. The client sampling rate is $p = 0.5$.

lution layers with 3 x 3 kernels, followed by an FC layer with 128 units and ReLu activation, and a final softmax output layer. For local training, we use noisy Adam with base learning rate 0.001.

Performance of Private Models. We test three values for the client sampling rate p: $0.25, 0.5$ and 1.0, and three values of noise scale $\sigma : 0.75, 0.9$, and 1.0. We use batch size $B = 16$ and run local training for $K = 38$ iterations between synchronization. The total number of samples processed between synchronization is 608, so we are approximately running local training for one epoch. Table 1 reports the synchronization rounds R and the privacy parameter $\mu_{\max}$, when the average prediction accuracy across 100 clients is above 90%. Table 1 also shows an intuitive phenomenon: a larger client sampling rate and a smaller noise level lead to faster convergence, see also Figure E.2. One might notice the privacy parameter is slighter larger than one usually see in a centralized training setting. Recall that Theorem 2 states that the privacy parameter μ_j scales linearly with a constant c_j. Loosely speaking, c_j is the product of the data sampling rate $\frac{B_j}{n_j}$ and the squared training iterations $\sqrt{KR}$: $\frac{B_j}{n_j}\sqrt{KR} \to c_j$ as $KR \to \infty$. In our simulated federated environment, each client holds only 1% data of the whole dataset, and the batch size is approximately 1/16 of the normal setting. Therefore, the effective data sampling rate B/n is much larger. Besides, it also takes much more iterations for the algorithm to converge in the federated setting. This leads to a larger privacy parameter. Interesting, there is also a trade-off between data sampling rate and computing iterations which might affect the privacy parameter, and we shall discuss this later in this section.

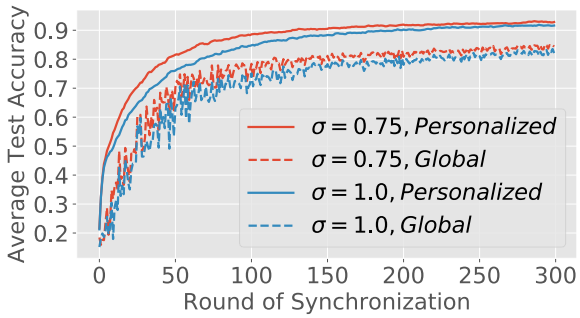

Figure 4.1: The personalized models outperform the global model in the MNIST experiments. The client sampling rate is $p = 0.5$.

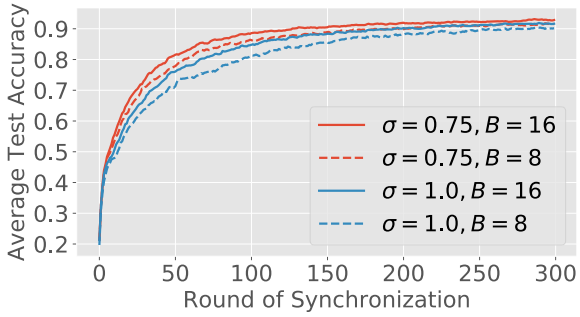

Figure 4.2: A larger batch size leads to faster convergence for the MNIST experiments. The client sampling rate is $p = 0.5$.

Accuracy Gain from Personalization. Figure 4.1 investigates the personalization performance of our approach. It compares the average test accuracy for the private global model and private personalized models. We plot the results when the noise scale $\sigma = 0.75$ and 1.0, where the client sampling

rate is $p = 0.5$. For both cases, personalized models significantly outperform the global model. The results for the other sampling rates are similar.

Privacy vss Computation Trade-off. As presented in Theorem 2, the privacy parameter μ_j scales linearly with a constant c_j. Informally, this is the product of the data sampling rate $\frac{B_j}{n_j}$ and the squared training iterations $\sqrt{KR}$. Since c_j scales linearly with B_j but sublinearly with K, using a smaller batch size would lead to a more private model if one processes the same amount of total data examples. For example, $\frac{B_j/2}{n_j}\sqrt{2KR} < \frac{B_j}{n_j}\sqrt{KR}$. However, the batch size has great impact on the rate of convergence. Figure 4.2 illustrates this phenomenon. Fixing the client sampling rate $p = 0.5$, we decrease the batch size from $B = 16$ to $B = 8$, and double the number of local iterations to $K = 76$. The small batch size ($B = 8$) runs take more rounds to achieve the same test accuracy, which means it processes more data examples in total. Table 2 demonstrates such a trade-off between privacy and computation. We compare the total number of training iterations KR, the per-client total training examples BKR, and the privacy parameter $\mu_{\max}$. As before, the results are reported when the average test accuracy achieves 90%. Compared with the large batch size ($B = 16$) runs, the small batch size ($B = 8$) runs obtain smaller privacy parameters, which are roughly 0.83× as obtained by the $B = 16$ runs. However, they take approximately 2.78× iterations to achieve 90% accuracy, which translates to 1.39× total number of samples.

4.2 Non-IID CIFAR

The CIFAR-10 dataset contains 50,000 training images and 10,000 test images of 10 classes. We use the same experiment setup as (Hsu et al., 2019). There are 100 clients, each holds 500 training images and 200 testing images. For each client, we generate data using the following probabilistic model:

1. Sample the class probability $q \sim \text{Dir}(\beta)$.
2. Sample $\theta^{\text{tr}} \sim \text{Multinomial}(q, 500)$.
3. Sample θ_i^{tr} images with label i from the training set without replacement.
4. Likewise, sample $\theta^{\text{te}} \sim \text{Multinomial}(q, 200)$ and the testing data accordingly.

The hyperparameter α controls the heterogeneity of the client data distributions. With $\beta \to \infty$, all the clients have identical class distributions; with $\beta \to 0$, the probability vector q will be one-hot and each client will hold samples from only one class. We use $\beta = 0.5$ throughout our experiments, see Figure E.3a for a visual illustration of the label proportions. Due to the GPU memory limit, we use the CNN model from the TensorFlow tutorial[3], like the previous work (McMahan et al., 2017; Hsu et al., 2019). This architecture is not state-of-the-art for CIFAR, but sufficient to demonstrate the relative performance for private and non-private models.

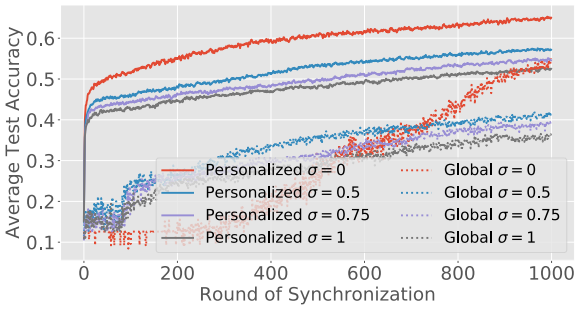

Figure 4.3: Average top-1 test accuracy vs. synchronization rounds for the CIFAR-10 experiments. The client sampling rate is $p = 0.5$.

p	σ	R	$\mu_{\max}$	Top-1 Acc	Non-Pri Acc
1.0	1.0	468	6.70	52.03	64.72
	0.75	321	9.77	52.22	62.55
	0.5	207	26.81	52.23	59.61
0.5	1.0	904	9.31	52.07	64.65
	0.75	671	14.13	52.04	62.53
	0.5	405	37.51	52.01	59.85

Table 3: CIFAR-10 experiment results: the round of synchronization and the corresponding privacy parameter when the average top-1 accuracy reaches 52%.

We observe that Adam is more stable than SGD for training private models, although SGD generalizes better on non-private models. Thereby, we train the models by noisy Adam with base learning 0.005 and weight decay 0.0005. The learning rate is decayed by a factor of 0.99 every 10 epochs. We use batch size $B = 16$ and run local training for $K = 32$ iterations. Figure 4.3 plots the top-1 test accuracy curve when the client samping rate $p = 0.5$. We can observe the privacy-accuracy trade-off: the test accuracy moderately decreases as the model becomes more private, i.e. trained with larger σ. Meanwhile, personalized models still outperform the global model. Table 3 reports the round of synchronization and corresponding privacy parameter when the average top-1 accuracy reaches 52%.

[3] https://www.tensorflow.org/tutorials/images/cnn.

Acknowledgments

We are grateful to Arun Kuchibhotla and Jinshuo Dong for insightful discussions. This work was supported in part by NIH through R01-GM124111 and RF1-AG063481, NSF through CAREER DMS-1847415, CCF-1763314, and CCF-1934876, a Facebook Faculty Research Award, and an Alfred Sloan Research Fellowship.

References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. (2016). Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318.

Balle, B., Barthe, G., Gaboardi, M., Hsu, J., and Sato, T. (2019). Hypothesis testing interpretations and renyi differential privacy. *arXiv preprint arXiv:1905.09982*.

Bu, Z., Dong, J., Long, Q., and Su, W. J. (2020). Deep learning with Gaussian differential privacy. *Harvard Data Science Review*, 2020(23).

Bun, M., Dwork, C., Rothblum, G. N., and Steinke, T. (2018). Composable and versatile privacy via truncated cdp. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 74–86.

Bun, M. and Steinke, T. (2016). Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography Conference*, pages 635–658. Springer.

Deng, Y., Kamani, M. M., and Mahdavi, M. (2020). Adaptive personalized federated learning. *arXiv preprint arXiv:2003.13461*.

Dinh, C. T., Tran, N. H., and Nguyen, T. D. (2020). Personalized federated learning with moreau envelopes. *arXiv preprint arXiv:2006.08848*.

Dong, J., Roth, A., and Su, W. J. (2019). Gaussian differential privacy. *To appear in Journal of the Royal Statistical Society: Series B (Statistical Methodology)*.

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

Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., and Naor, M. (2006a). Our data, ourselves: Privacy via distributed noise generation. In *Proceedings of the 24th Annual International Conference on The Theory and Applications of Cryptographic Techniques*, EUROCRYPT'06, pages 486–503, Berlin, Heidelberg. Springer-Verlag.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. (2006b). Calibrating noise to sensitivity in private data analysis. In *Proceedings of the Third Conference on Theory of Cryptography*, TCC'06, pages 265–284, Berlin, Heidelberg. Springer-Verlag.

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

Dwork, C. and Rothblum, G. N. (2016). Concentrated differential privacy. *arXiv preprint arXiv:1603.01887*.

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

Fredrikson, M., Jha, S., and Ristenpart, T. (2015). Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333.

Geyer, R. C., Klein, T., and Nabi, M. (2017). Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*.

Hanzely, F. and Richtárik, P. (2020). Federated learning of a mixture of global and local models. *arXiv preprint arXiv:2002.05516*.

Hsu, J., Huang, Z., Roth, A., Roughgarden, T., and Wu, Z. S. (2016a). Private matchings and allocations. *SIAM Journal on Computing*, 45(6):1953–1984.

Hsu, J., Huang, Z., Roth, A., and Wu, Z. S. (2016b). Jointly private convex programming. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 580–599. SIAM.

Hsu, T.-M. H., Qi, H., and Brown, M. (2019). Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*.

Huang, Y., Chu, L., Zhou, Z., Wang, L., Liu, J., Pei, J., and Zhang, Y. (2020). Personalized federated learning: An attentive collaboration approach. *arXiv preprint arXiv:2007.03797*.

Kairouz, P., Oh, S., and Viswanath, P. (2017). The composition theorem for differential pri-

vacy. *IEEE Transactions on Information Theory*, 63(6):4037–4049.

Kasiviswanathan, S. P., Lee, H. K., Nissim, K., Raskhodnikova, S., and Smith, A. (2011). What can we learn privately? *SIAM Journal on Computing*, 40(3):793–826.

Kearns, M., Pai, M., Roth, A., and Ullman, J. (2014). Mechanism design in large games: Incentives and privacy. In *Proceedings of the 5th conference on Innovations in theoretical computer science*, pages 403–410.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Konečnỳ, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. (2016). Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*.

Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images.

LeCun, Y. (1998). The mnist database of handwritten digits.

Li, J., Khodak, M., Caldas, S., and Talwalkar, A. (2019). Differentially private meta-learning. *arXiv preprint arXiv:1909.05830*.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. (2018). Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*.

Liu, C., He, X., Chanyaswad, T., Wang, S., and Mittal, P. (2019). Investigating statistical privacy frameworks from the perspective of hypothesis testing. *Proceedings on Privacy Enhancing Technologies*, 2019(3):233–254.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., and Aguera y Arcas, B. (2017). Communication-efficient learning of deep networks from decentralized data. In *AISTATS*.

McMahan, H. B., Ramage, D., Talwar, K., and Zhang, L. (2018). Learning differentially private recurrent language models. In *ICLR*.

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

Murtagh, J. and Vadhan, S. (2016). The complexity of computing the optimal composition of differential privacy. In *Theory of Cryptography Conference*, pages 157–175. Springer.

Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151.

Shokri, R., Stronati, M., Song, C., and Shmatikov, V. (2017). Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE.

Singh, I., Zhou, H., Yang, K., Ding, M., Lin, B., and Xie, P. (2020). Differentially-private federated neural architecture search. *arXiv preprint arXiv:2006.10559*.

Truex, S., Baracaldo, N., Anwar, A., Steinke, T., Ludwig, H., Zhang, R., and Zhou, Y. (2019). A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, pages 1–11.

Wasserman, L. and Zhou, S. (2010). A statistical framework for differential privacy. *Journal of the American Statistical Association*, 105(489):375–389.

Zheng, Q., Dong, J., Long, Q., and Su, W. J. (2020). Sharp composition bounds for Gaussian differential privacy via Edgeworth expansion. In *International Conference on Machine Learning*, pages 11420–11435.