Constrained Differentially Private Federated Learning for Low-bandwidth Devices

Raouf Kerkouche[1] **Gergely Ács**[2] **Claude Castelluccia**[1] **Pierre Genevès**[3]

[1]Privatics team, Univ. Grenoble Alpes, Inria, 38000, Grenoble, France
[2]Crysys Lab, BME-HIT
[3]Tyrex team, Univ. Grenoble Alpes, CNRS, Inria,, Grenoble INP, LIG

Abstract

Federated learning becomes a prominent approach when different entities want to learn collaboratively a common model without sharing their training data. However, Federated learning has two main drawbacks. First, it is quite bandwidth inefficient as it involves a lot of message exchanges between the aggregating server and the participating entities. This bandwidth and corresponding processing costs could be prohibitive if the participating entities are, for example, mobile devices. Furthermore, although federated learning improves privacy by not sharing data, recent attacks have shown that it still leaks information about the training data.
This paper presents a novel privacy-preserving federated learning scheme. The proposed scheme provides theoretical privacy guarantees, as it is based on Differential Privacy. Furthermore, it optimizes the model accuracy by constraining the model learning phase on few selected weights. Finally, as shown experimentally, it reduces the upstream *and* downstream bandwidth by up to 99.9% compared to standard federated learning, making it practical for mobile systems.

1 INTRODUCTION

In Machine Learning, different entities may want to collaborate in order to improve their local model accuracy. In traditional machine learning, such collaboration requires to first store all entities' data on a centralized server and then to train a model on it. Such data centralization might be problematic when the data are sensitive and data privacy is required. In order to mitigate this problem, Federated learning, which allows different entities to learn collaboratively a common model without sharing their data, was introduced [Shokri and Shmatikov, 2015, McMahan et al., 2016]. Instead of sharing the training data, Federated Learning shares the model parameters between a server, which plays the role of aggregator, and the participating entities. Although Federated Learning improves privacy, model parameters can leak information about the training data. Indeed, Zhu et al. [2019], Zhao et al. [2020], Geiping et al. [2020] presented some attacks that allow an adversary to reconstruct pieces of the training data of some entities. Nasr et al. [2019] define a membership attack that allows to infer if a particular record is included in the data of a specific entity. Similarly, Melis et al. [2018] define an attack which aims at inferring if a subgroup of people with a specific property, like for example skin color or ethnicity, is included in the dataset of a particular participating entity. A solution to prevent these attacks and provide theoretical guarantees in to use a privacy model called Differential Privacy [Dwork and Roth, 2014]. Differential Privacy has been applied to federated learning in order to protect either each record included in the dataset of any entity (record-level guarantee), or the whole dataset of any entity (client-level guarantee). Unfortunately, it is well-known that Differential Privacy drastically degrades the accuracy of the global model as it requires to add random noise to the gradients (record-level) or to the updates (client-level) of each client. Recent work by Kerkouche et al. [2020] shows that this accuracy penalty can be reduced if the model is compressed, as compression reduces the required amount of noise. Furthermore, Kerkouche et al. [2020] show that accuracy can be further improved by adding noise only to the largest update's values as adding noise on values close to 0 is likely to lead to random update values.

Following up on these results, we propose a novel differentially private federated learning solution that improves the model accuracy (1) by updating only a fixed subset of the model weights, and (2) by maintaining the other weights constant. The proposed scheme provides theoretical privacy guarantees, as it is based on Differential Privacy. Furthermore, it optimizes the model accuracy by constraining the model learning phase on a few selected weights. As all participants always update the same set of weights and transfer

Accepted for the 37th Conference on Uncertainty in Artificial Intelligence (UAI 2021).

them to the server for aggregation, the proposal can be easily integrated with secure aggregation [Bonawitz et al., 2016], which allows parties to add less noise than other decentralized perturbation approaches such as randomized response [Erlingsson et al., 2014] used in local differential privacy. Moreover, it also reduces the upstream and downstream bandwidth by a factor of 1000 compared to standard federated learning, making it practical for mobile systems.

The paper is structured as follows: In Section 2 we introduce the necessary background to understand the proposal, in Section 3 we define our solution called FL-TOP and in Section 3.2 its private extension called FL-TOP-DP.

2 BACKGROUND

2.1 FEDERATED LEARNING (FL-STD)

In federated learning [Shokri and Shmatikov, 2015, McMahan et al., 2016], multiple parties (clients) build a common machine learning model from union of their training data without sharing them with each other. At each round of the training, a selected set of clients retrieve the global model from the parameter server, update the global model based on their own training data, and send back their updated model to the server. The server aggregates the updated models of all clients to obtain a global model that is re-distributed to some selected parties in the next round.

In particular, a subset $\mathbb{K}$ of all N clients are randomly selected at each round to update the global model, and $C = |\mathbb{K}|/N$ denotes the fraction of selected clients. At round t, a selected client $k \in \mathbb{K}$ executes T_{gd} local gradient descent iterations on the common model $\mathbf{w}_{t-1}$ using its own training data D_k ($D = \cup_{k\in\mathbb{K}} D_k$), and obtains the updated model $\mathbf{w}_t^k$, where the number of weights is denoted by n (i.e., $|\mathbf{w}_t^k| = |\Delta\mathbf{w}_t^k| = n$ for all k and t). Each client k submits the update $\Delta\mathbf{w}_t^k = \mathbf{w}_t^k - \mathbf{w}_{t-1}^k$ to the server, which then updates the common model as follows: $\mathbf{w}_t = \mathbf{w}_{t-1} + \sum_{k\in\mathbb{K}} \frac{|D_k|}{\sum_j |D_j|} \Delta\mathbf{w}_t^k$, where $|D_k|$ is known to the server for all k (a client's update is weighted with the size of its training data). The server stops training after a fixed number of rounds T_{cl}, or when the performance of the common model does not improve on a held-out data.

Note that each D_k may be generated from different distributions (i.e., non-IID case), that is, any client's local dataset may not be representative of the population distribution [McMahan et al., 2016]. This can happen, for example, when not all output classes are represented in every client's training data. The federated learning of neural networks is summarized in Alg. 1. In the sequel, each client is assumed to use the same model architecture.

The motivation of federated learning is three-fold: first, it aims to provide confidentiality of each participant's training data by sharing only model updates instead of potentially sensitive training data. Second, in order to decrease communication costs, clients can perform multiple local SGD iterations before sending their update back to the server. Third, in each round, only a few clients are required to perform local training of the common model, which further diminishes communication costs and makes the approach especially appealing with large number of clients.

However, several prior works have demonstrated that model updates do leak potentially sensitive information [Nasr et al., 2019, Melis et al., 2018]. Hence, simply not sharing training data *per se* is not enough to guarantee their confidentiality.

2.2 DIFFERENTIAL PRIVACY

Differential privacy allows a party to privately release information about a dataset: a function of an input dataset is perturbed, so that any information which can differentiate a record from the rest of the dataset is bounded Dwork and Roth [2014].

Definition 1 (Privacy loss)**.** *Let $\mathcal{A}$ be a privacy mechanism which assigns a value $Range(\mathcal{A})$ to a dataset D. The privacy loss of $\mathcal{A}$ with datasets D and D' at output $O \in Range(\mathcal{A})$ is a random variable $\mathcal{P}(\mathcal{A}, D, D', O) = \log \frac{\Pr[\mathcal{A}(D)=O]}{\Pr[\mathcal{A}(D')=O]}$ where the probability is taken on the randomness of $\mathcal{A}$.*

Definition 2 ((ϵ, δ)-Differential Privacy [Dwork and Roth, 2014])**.** *A privacy mechanism $\mathcal{A}$ guarantees (ε, δ)-differential privacy if for any database D and D', differing on at most one record, $\Pr_{O\sim\mathcal{A}(D)}[\mathcal{P}(\mathcal{A}, D, D', O) > \varepsilon] \leq \delta$.*

Intuitively, this guarantees that an adversary, provided with the output of $\mathcal{A}$, can draw almost the same conclusions (up to ε with probability larger than $1 - \delta$) about any record no matter if it is included in the input of $\mathcal{A}$ or not. That is, for any record owner, a privacy breach is unlikely to be due to its participation in the dataset.

Moments Accountant. Differential privacy maintains composition; the privacy guarantee of the k-fold adaptive composition of $\mathcal{A}_{1:k} = \mathcal{A}_1, \ldots, \mathcal{A}_k$ can be computed using the moments accountant method Abadi et al. [2016]. In particular, it follows from Markov's inequality that $\Pr[\mathcal{P}(\mathcal{A}, D, D', O) \geq \varepsilon] \leq \mathbb{E}[\exp(\lambda\mathcal{P}(\mathcal{A}, D, D', O))]/\exp(\lambda\varepsilon)$ for any output $O \in Range(\mathcal{A})$ and $\lambda > 0$. $\mathcal{A}$ is (ε, δ)-DP with $\delta = \min_\lambda \exp(\alpha_\mathcal{A}(\lambda) - \lambda\varepsilon)$, where $\alpha_\mathcal{A}(\lambda) = \max_{D,D'} \log \mathbb{E}_{O\sim\mathcal{A}(D)}[\exp(\lambda\mathcal{P}(\mathcal{A}, D, D', O))]$ is the log of the moment generating function of the privacy loss. The privacy guarantee of the composite mechanism $\mathcal{A}_{1:k}$ can be computed using that $\alpha_{\mathcal{A}_{1:k}}(\lambda) \leq \sum_{i=1}^k \alpha_{\mathcal{A}_i}(\lambda)$ Abadi et al. [2016].

Gaussian Mechanism. A fundamental concept of all DP sanitization techniques is the *global sensitivity* of a function [Dwork and Roth, 2014].

Definition 3 (Global L_p-sensitivity)**.** *For any function $f : \mathcal{D} \rightarrow \mathbb{R}^n$, the L_p-sensitivity of f is $\Delta_p f = \max_{D,D'} ||f(D) - f(D')||_p$, for all D, D' differing in at most one record, where $|| \cdot ||_p$ denotes the L_p-norm.*

The Gaussian Mechanism [Dwork and Roth, 2014] consists of adding Gaussian noise to the true output of a function. In particular, for any function $f : \mathcal{D} \rightarrow \mathbb{R}^n$, the Gaussian mechanism is defined as adding i.i.d Gaussian noise with variance $(\Delta_2 f \cdot \sigma)^2$ and zero mean to each coordinate value of $f(D)$. Recall that the pdf of the Gaussian distribution with mean μ and variance ξ^2 is $\mathsf{pdf}_{\mathcal{G}(\mu,\xi)}(x) = \frac{1}{\sqrt{2\pi}\xi} \exp\left(-\frac{(x-\mu)^2}{2\xi^2}\right)$.

In fact, the Gaussian mechanism draws vector values from a multivariate spherical (or isotropic) Gaussian distribution which is described by random variable $\mathcal{G}(f(D), \Delta_2 f \cdot \sigma \mathbf{I}_n)$, where n is omitted if its unambiguous in the given context.

3 FEDERATED PRUNING

In the standard federated learning scheme (FL-STD, in Section 2), the server sends the latest updated model to a randomly selected set of clients (downstream), and each client sends back its complete model update after local training to the server (upstream) at each round. Knowing that a model has on average millions of parameters (each is a floating point value represented on 32 bits), the network can suffer from large traffic both upstream and downstream.

Our solution, called FL-TOP, aims to reduce the large amount of network traffic by reducing both downstream and upstream traffic. Moreover, a privacy-preserving extension of this scheme, called FL-TOP-DP, is also proposed, which provides Differential Privacy for the whole training data of every client.

In what follows, we first describe the non-private scheme FL-TOP and then the privacy-preserving FL-TOP-DP.

3.1 FL-TOP: FEDERATED PRUNING FOR COMPRESSION

FL-TOP is inspired by the pruning techniques proposed in Han et al. [2016] (see Section 5 for more details), and it aims to reduce the amount of parameters exchanged downstream (from the server to the participating entities) and upstream (from the participating entities to the server). In our scheme, each client updates only a small subset, Top-K, of the model parameters (weights) at each round. Only the K weight values of these Top-K parameters are updated during training, and neither the clients nor the server need to transfer the values of the remaining $n - K$ parameters, where n is the total number of parameters. The set of non Top-K parameters do not change over the whole training and are identical for all clients. We experimentally show in Section 4 that, if these K parameters are chosen carefully, the performance penalty is negligible even if $K = 0.005 \cdot n$, that is, 99.5% of the model parameters are pruned. Note that unlike standard pruning techniques, where the set of pruned weights are re-selected after each SGD iteration [Han et al., 2016], our scheme always updates the same K parameters.

These Top-K parameters are selected by the server at the beginning of the protocol. More specifically, the server initializes the model and trains that with some public data that have a similar distribution as the clients' training data. After a few SGD iterations, the server selects the K parameters which values changed the most.

FL-TOP is described in Alg. 1. First, the server uses public data to identify the set $\mathbb{T}$ of the Top-K parameters ($K = |\mathbb{T}|$), before starting federated learning. In particular, starting from a public model $\mathbf{w}_0$, it accumulates the absolute value of gradients per parameter over T_{init} SGD iterations, and selects the K parameters with the largest accumulated gradients. After that, the values/updates[1] of these parameters are the only ones exchanged during the rest of the training between the server and the clients.

At each round, each selected client k uses the K updated weights $\hat{\mathbf{w}}_{t-1}$ received from the server to create a new weight vector $\mathbf{w}^k_{t-1}$ of size n, such that $\mathbf{w}^k_{t-1}$ is composed from the compressed vector $\hat{\mathbf{w}}^k_{t-1}$ of size $K \leq n$ (with coordinates in $\mathbb{T}$) and $n - K$ weights from the initialization vector $\mathbf{w}_0$. $\mathbf{w}_0$ is identical for all participants and can be generated from a shared seed. Note that when $K = |\mathbb{T}| = n$, the scheme is equivalent to FL-STD. The weight vector $\mathbf{w}^k_{t-1}$ is used to train the client's model. However, only the weights in $\mathbb{T}$ are updated while the remaining ones are kept fixed. To do that, the weights not in $\mathbb{T}$ are reinitialized after each SGD iteration to $\mathbf{w}_0$. The server receives only the values from $\mathbf{w}^k_t - \mathbf{w}^k_{t-1}$ at coordinates $\mathbb{T}$, denoted by $\mathcal{C}(\mathbf{w}^k_t - \mathbf{w}^k_{t-1})$ for short, from every client k, and updates the common model $\mathbf{w}_t$ with the average of these compressed updates (in Line 12).

3.2 FL-TOP-DP: DIFFERENTIALLY PRIVATE FEDERATED PRUNING

3.2.1 Privacy Model

We consider an adversary, or a set of colluding adversaries, who can access any update vector sent by the server or any clients at each round of the protocol. A plausible adversary is a participating entity, i.e. a malicious client or server, that wants to infer the training data used by other participants.

[1] weight values for downstream and update/gradients for upstream traffic

The adversary is *passive* (i.e., honest-but-curious), that is, it follows the learning protocol faithfully.

Different privacy requirements can be considered depending on what information the adversary aims to infer. In general, private information can be inferred about:

- any record (user) in any dataset of any client (*record-level privacy*),
- any client/party (*client-level privacy*).

To illustrate the above requirements, suppose that several banks build a common model to predict the creditworthiness of their customers. A bank certainly does not want other banks to learn the financial status of any of their customers (record privacy) and perhaps not even the average income of all their customers (client privacy).

Record-level privacy is a standard requirement used in the privacy literature and is usually weaker than client-level privacy. Indeed, client-level privacy requires to hide any information which is unique to a client including perhaps all its training data.

We aim at developing a solution that provides *client-level privacy and is also bandwidth efficient*. For example, in the scenario of collaborating banks, we aim at protecting any information that is unique to each single bank's training data. The adversary should not be able to learn from the received model or its updates whether any client's data is involved in the federated run (up to ε and δ). We believe that this adversarial model is reasonable in many practical applications when the confidential information spans over multiple samples in the training data of a single client (e.g., the presence of a group a samples, such as people from a certain race). Differential Privacy guarantees plausible deniability not only to any groups of samples of a client but also to any client in the federated run. Therefore, any negative privacy impact on a party (or its training samples) cannot be attributed to their involvement in the protocol run.

3.2.2 Operation

FL-TOP-DP is described in Alg. 3 is very similar to FL-TOP except that each client adds Gaussian noise to its Top-K model updates to guarantee client-level DP, and applies secure aggregation allowing the server to learn only the aggregated (and noisy) model update. More specifically, each client first calculates its compressed model update $\Delta \mathbf{w}_t^k = \mathcal{C}(\mathbf{w}_t^k - \mathbf{w}_{t-1}^k)$ (in Line 25) which is then clipped (in Line 26) to obtain $\Delta \hat{\mathbf{w}}_t^k$ with L_2-norm at most S. After that, random noise $\mathbf{z}_k \sim \mathcal{G}(0, S\sigma\mathbf{I}/\sqrt{\mathbb{K}})$ is added to $\Delta \hat{\mathbf{w}}_t^k$ such that the sum $\sum_{k\in\mathbb{K}}(\Delta \hat{\mathbf{w}}_t^k + \mathbf{z}_k) = \sum_{k\in\mathbb{K}} \Delta \hat{\mathbf{w}}_t^k + \mathcal{G}(0, S\sigma\mathbf{I})$ as the sum of Gaussian random variables also follows Gaussian distribution[2] and then differential privacy is satisfied where ε and δ can be computed using the moments accountant described in Section 2.2. Recall that the Top-K coordinates in $\mathbb{T}$ are selected and distributed by the server, which is honest-but-curious by assumption.

However, as the noise is inversely proportional to $\sqrt{\mathbb{K}}$, $\mathbf{z}_k$ is likely to be small if $|\mathbb{K}|$ is too large. Therefore, the adversary accessing an individual update $\Delta \hat{\mathbf{w}}_t^k + \mathbf{z}_k$ can almost learn a non-noisy update since $\mathbf{z}_k$ is small. Hence, each client uses secure aggregation to encrypt its individual update before sending it to the server. Upon reception, the server sums the encrypted updates as:

$$\sum_{k\in\mathbb{K}} \mathbf{c}_t^k = \sum_{k\in\mathbb{K}} \mathsf{Enc}_{K_k}(\Delta \hat{\mathbf{w}}_t^k + \mathbf{z}_k) = \sum_{k\in\mathbb{K}} \Delta \hat{\mathbf{w}}_t^k + \sum_{k\in\mathbb{K}} \mathbf{z}_k$$
$$= \sum_{k\in\mathbb{K}} \Delta \hat{\mathbf{w}}_t^k + \mathcal{G}(0, S\sigma\mathbf{I}) \quad (1)$$

where $\mathsf{Enc}_{K_k}(\Delta \hat{\mathbf{w}}_t^k + \mathbf{z}_k) = \Delta \hat{\mathbf{w}}_t^k + \mathbf{z}_k + \mathbf{K}_k \mod p$ and $\sum_k \mathbf{K}_k = 0$ (see Ács and Castelluccia [2011], Bonawitz et al. [2016] for details). Here the modulo is taken element-wise and $p = 2^{\lceil \log_2(\max_k ||\Delta \hat{\mathbf{w}}_t^k + \mathbf{z}_k||_\infty |\mathbb{K}|)\rceil}$. Let $\gamma_t^k = 1/\max\left(1, \frac{||\Delta \mathbf{w}_t^k||_2}{S}\right)$. Then,

$$\sum_{k\in\mathbb{K}} \Delta \hat{\mathbf{w}}_t^k = \sum_{k\in\mathbb{K}} \gamma_t^k \Delta \mathbf{w}_t^k = \sum_{k\in\mathbb{K}} \gamma_t^k \mathcal{C}(\mathbf{w}_t^k - \mathbf{w}_{t-1}^k, \mathbb{T})$$
$$= \mathcal{C}(\sum_{k\in\mathbb{K}} \gamma_t^k (\mathbf{w}_t^k - \mathbf{w}_{t-1}^k), \mathbb{T}) \quad (2)$$

where the last equality comes from the linearity of the compression operation. Indeed, recall that each client selects the values of the *same* Top-K coordinates from $\mathbb{T}$. Plugging Eq. (2) into Eq. (1). we get that

$$\sum_{k\in\mathbb{K}} \mathbf{c}_t^k = \mathcal{C}(\sum_{k\in\mathbb{K}} \gamma_t^k (\mathbf{w}_t^k - \mathbf{w}_{t-1}^k), \mathbb{T}) + \mathcal{G}(0, S\sigma\mathbf{I})$$

Privacy analysis: The server can only access the noisy aggregate which is sufficiently perturbed to ensure differential privacy; any client-specific information that could be inferred from the noisy aggregate is tracked and quantified by the moments accountant, described in Section 2.2, as follows.

Let $\eta_0(x|\xi) = \mathsf{pdf}_{\mathcal{G}(0,\xi)}(x)$ and $\eta_1(x|\xi) = (1-C)\mathsf{pdf}_{\mathcal{G}(0,\xi)}(x) + C\mathsf{pdf}_{\mathcal{G}(1,\xi)}(x)$ where C is the sampling probability of a single client in a single round. Let $\alpha(\lambda|C) = \log\max(E_1(\lambda,\xi,C), E_2(\lambda,\xi,C))$ where $E_1(\lambda,\xi,C) = \int_{\mathbb{R}} \eta_0(x|\xi,C) \cdot \left(\frac{\eta_0(x|\xi,C)}{\eta_1(x|\xi,C)}\right)^\lambda dx$ and $E_2(\lambda,\xi,C) = \int_{\mathbb{R}} \eta_1(x|\xi,C) \cdot \left(\frac{\eta_1(x|\xi,C)}{\eta_0(x|\xi,C)}\right)^\lambda dx$.

Theorem 1 (Privacy of FL-TOP-DP)**.** *FL-TOP-DP is* $(\min_\lambda(T_{\mathsf{cl}} \cdot \alpha(\lambda|C) - \log\delta)/\lambda, \delta)$*-DP.*

Given a fixed value of δ, ε is computed numerically as in Abadi et al. [2016], Mironov et al. [2019].

[2] More precisely, $\sum_i \mathcal{G}(\nu_i, \xi_i) = \mathcal{G}(\sum_i \nu_i, \sqrt{\sum_i \xi_i^2})$

Algorithm 1: FL-TOP: Federated Learning

Server:
  Initialize common model w_0
  Select set $\mathbb{T}$ of Top-K updated weights' coordinates via public dataset
  for $t = 1$ **to** T_{cl} **do**
    Select $\mathbb{K}$ clients uniformly at random
    for *each client* k *in* $\mathbb{K}$ **do**
      $\mathbf{c}_t^k = \mathbf{Client}_k(\mathcal{C}(\mathbf{w}_{t-1}, \mathbb{T}))$
    end
    $\mathbf{w}_t = \mathbf{w}_0$
    $j = 1$
    for *each coordinate* i *in* $\mathbb{T}$ **do**
      $\mathbf{w}_t[i] = \mathbf{w}_{t-1}[i] + \sum_k \frac{\mathbf{c}_t^k[j]}{|\mathbb{K}|}$
      $j = j + 1$
    end
  end
  Output: Global model $\mathbf{w}_t$

$\mathbf{Client}_k(\hat{\mathbf{w}}_{t-1}^k)$**:**
  $\mathbf{w}_{t-1}^k = \mathbf{w}_0$
  $j = 1$
  for *each coordinate* i *in* $\mathbb{T}$ **do**
    $\mathbf{w}_{t-1}^k[i] = \hat{\mathbf{w}}_{t-1}^k[j]$
    $j = j + 1$
  end
  $\mathbf{w}_t^k = \mathbf{Top_k SGD}(D_k, \mathbf{w}_{t-1}^k, \mathbf{w}_0, T_{\text{gd}}, \mathbb{T})$
  Output: Model update $\mathcal{C}(\mathbf{w}_t^k - \mathbf{w}_{t-1}^k, \mathbb{T})$

Algorithm 2: $\mathbf{Top_k}$-Stochastic Gradient Descent

Input: D : training data, T_{gd} : local epochs, $\mathbf{w}$: weights, $\mathbf{w}_0$: first weights' initialization, $\mathbb{T}$: set of Top-K values coordinates .

for $t = 1$ **to** T_{gd} **do**
  Select batch $\mathbb{B}$ from D randomly
  $\mathbf{u} = -\eta \nabla f(\mathbb{B}; \mathbf{w})$
  for *each coordinate* i *in* $\mathbb{T}$ **do**
    $\mathbf{w}[i] = \mathbf{w}[i] + \mathbf{u}[i]$
  end
end
Output: Model $\mathbf{w}$

3.2.3 Remarks

The magnitude of the added Gaussian noise is proportional to the clipping threshold S, which is in turn calibrated to the norm of the model update. However, the norm of the model update increases if the model size increases [Zhu et al., 2020], and hence S should be chosen sufficiently large to guarantee fast convergence with large accuracy. On the other hand, too large S also increases the perturbation error caused by the added noise.

FL-TOP aims to diminish this perturbation error by reducing S via compression which also increases the L_2-norm of the compressed update vector. This is illustrated in Figure 1, which shows that the norm of the Top-K coordinates with FL-TOP tend to be larger than with FL-STD (i.e., when all coordinates get updated not only the Top-K). Therefore, besides decreasing the magnitude of the added noise, FL-TOP also decreases the relative error on the retained parameters. These together decrease the perturbation error caused by the added noise.

Notice that there exist other alternatives to identify the Top-K coordinates in a privacy-preserving manner than using a public dataset. For example, every client can select the Top-K parameters with the largest magnitude during the first rounds locally, and send them to the server for aggregation. More specifically, each client creates a parameter vector with size n, where the Top-K coordinates are set to 1 while the rest are kept 0. Then, these binary vectors are noised and aggregated by the server like in Section 3.2.2. In the rest of the training, all participants exchange only the updates and weights of the these Top-K parameters like in FL-TOP. However, aside from consuming more privacy budget, this approach also has lower accuracy than our proposal according to our tests. Moreover, it has larger communication cost in the initialization phase when the Top-K parameters are identified and the whole binarized parameter vector is sent for aggregation.

4 EXPERIMENTAL RESULTS

The goal of this section is to evaluate the performance of our proposed schemes FL-TOP and FL-TOP-DP on a benchmark dataset and a realistic in-hospital mortality prediction scenario. We aim at evaluating their performance with different levels of compression and comparing them with the performance of the following learning protocols[3]:

- FL-STD: The Standard Federated Learning scheme as described in Section 2.1 (see Alg. 1).
- FL-BASIC: A Federated Learning scheme that updates a random subset of parameters instead of the Top-K parameters at each SGD iteration. This subset is re-selected at the beginning of each new round. The $n-k$ non-selected parameters are still reinitialized after each SGD update as in FL-TOP.
- FL-CS: A Federated Learning scheme that uses Compressive sensing (CS) to compress model updates from Kerkouche et al. [2020]. See Section 5 for more details.

Note that all compression operators in the baselines are linear (just like FL-TOP-DP), and hence they can also be used with secure aggregation. Similarly to FL-TOP-DP, the

[3]More baselines are considered but due to the lack of space, we have decided to present only those which return the best results. All other results can be found in the appendix(Section D).

Algorithm 3: FL-TOP-DP: Federated Learning

Server:
  Initialize common model w_0
  Select set $\mathbb{T}$ of Top-K updated weights' coordinates via public dataset
  for $t = 1$ **to** T_{cl} **do**
    Select $\mathbb{K}$ clients uniformly at random
    for *each client* k *in* $\mathbb{K}$ **do**
      $\mathbf{c}_t^k = \mathbf{Client}_k(\mathcal{C}(\mathbf{w}_{t-1}, \mathbb{T}))$
    end
    $\mathbf{w}_t = \mathbf{w}_0$
    $j = 1$
    for *each coordinate* i *in* $\mathbb{T}$ **do**
      $\mathbf{w}_t[i] = \mathbf{w}_{t-1}[i] + \sum_k \frac{\mathbf{c}_t^k[j]}{|\mathbb{K}|}$
      $j = j + 1$
    end
  end
  Output: Global model $\mathbf{w}_t$

$\mathbf{Client}_k(\hat{\mathbf{w}}_{t-1}^k)$**:**
  $\mathbf{w}_{t-1}^k = \mathbf{w}_0$
  $j = 1$
  for *each coordinate* i *in* $\mathbb{T}$ **do**
    $\mathbf{w}_{t-1}^k[i] = \hat{\mathbf{w}}_{t-1}^k[j]$
    $j = j + 1$
  end
  $\mathbf{w}_t^k = \mathbf{Top_k SGD}(D_k, \mathbf{w}_{t-1}^k, \mathbf{w}_0, T_{\mathsf{gd}}, \mathbb{T})$
  $\Delta\mathbf{w}_t^k = \mathcal{C}(\mathbf{w}_t^k - \mathbf{w}_{t-1}^k, \mathbb{T})$
  $\Delta\hat{\mathbf{w}}_t^k = \Delta\mathbf{w}_t^k / \max\left(1, \frac{||\Delta\mathbf{w}_t^k||_2}{S}\right)$
  Output: $\mathsf{Enc}_{K_k}(\mathcal{G}(\Delta\hat{\mathbf{w}}_t^k, S\mathbf{I}\sigma/\sqrt{|K|}))$

private extensions (i.e., FL-STD-DP, FL-BASIC-DP and FL-CS-DP) also clip and then noise the compressed updates.

We evaluate the above learning algorithms on the well-known Fashion-MNIST dataset [Xiao et al., 2017] and on the Premier Healthcare Database, which is a real-world medical dataset of 1.2 million of US hospital patients [4]. More details can be found in Appendix A.1 and Appendix B.1.

Recall that the Top-K weights are selected before starting the federated learning process using public data. For Fashion-MNIST, we randomly select a batch with size 10 from MNIST dataset [LeCun and Cortes, 2010] described in Appendix B.2. For the medical dataset, we did not find any public dataset with the same features as ours, and for this reason, we selected randomly from the dataset a batch of 356 patients[5]. This set is used only by the server and never by any client. Afterwards, the server performs T_{init} SGD iterations starting from the model parameters $\mathbf{w}_0$ on the same batch to identify the Top-K weights. We experimentally show later that even these small batches are enough for the server to find a good set of Top-K weights.

In order to select the clipping threshold S, the server executes a single training round locally, which is composed of T_{gd} SGD iterations starting from the model parameters $\mathbf{w}_0$, using the batch from the public data. The clipping threshold S is set to the L_2-norm of the Top-K weight update obtained for this single training round. For FL-BASIC-DP, the same steps are repeated for 100 times, where a new random set of trainable weights with size K are selected each time, which yields 100 L_2-norm values. S is set to the median of these L_2-norm values. We think that this approach is more fair, because the set of trainable weights is re-selected at each round in FL-BASIC-DP. The computed values of S can be found in Table 4 and Table 5 for Fashion-MNIST and Medical dataset, respectively. More information about the model architectures and the hyper-parameter selection can be found in Appendix A.

4.1 RESULTS

Figure 1 displays the distribution of the Top-K updated weights for FL-TOP and FL-STD at the end of the training. We select the weights when each scheme reached the best accuracy over 200 and best balanced accuracy[6] over 100 rounds for fashion-MNIST and the medical dataset, respectively. We choose the smallest compression ratio r that leads to the best accuracy for the FL-TOP-DP scheme. Table 1 shows that FL-TOP-DP reaches the best accuracy, 0.81, when $r = 0.5\%$ on fashion-MNIST and reaches the best accuracy, 0.69, when $r = 0.1\%$ on the medical dataset. Both figures validate the intuition that by constraining the model to update only a small set K of the total weights, these Top-K become more important and reach larger values. This result is important when differential privacy is used as it leads to larger value-to-noise level and therefore better performance.

Table 1 represents the best accuracy over 200 rounds for each scheme on the Fashion-MNIST dataset. $Round$ corresponds to the round when the best accuracy is reached and $Cost$ is the average bandwidth consumption calculated as: $r \times n \times 32 \times Round \times C$, where 32 is the number of bits necessary to represent a float value, n is the uncompressed model size, $r = \frac{|\mathbb{T}|}{n}$, $|\mathbb{T}|$ is the compressed model size, C is the sampling probability of a client, and $Round$ is the round when we get the the best accuracy.

Table 2 represents the best balanced accuracy over 100 rounds for each scheme on the Medical dataset. $AUROC$ (area under the receiver operating characteristic curve - see Appendix A.4) corresponds to the $AUROC$ value when the best balanced accuracy is reached.

[4] https://www.premierinc.com/newsroom/education/premier-healthcare-database-whitepaper

[5] Reduced to 24 patients when we train via downsampling with 12 patients for each class

[6] See Appendix A.4 for more details.

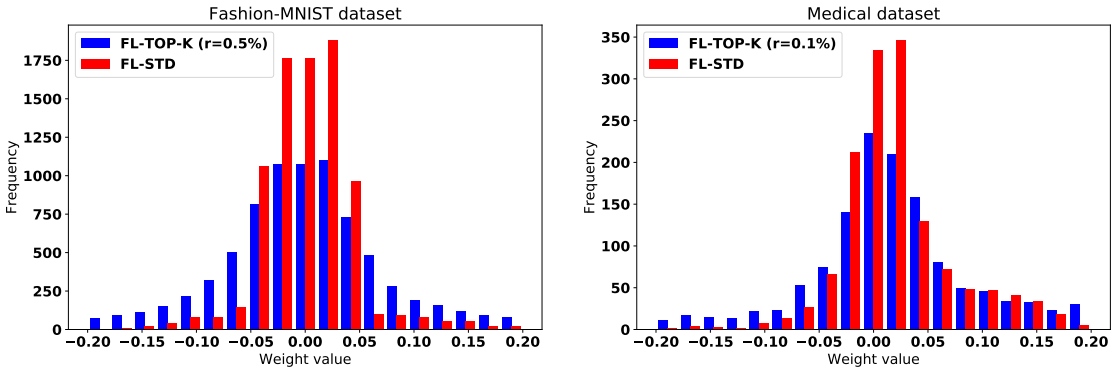

Figure 1: Distributions of the Top-K weight values (after convergence) for both FL-TOP and FL-STD schemes with the Fashion-MNIST dataset (left) and the medical dataset (right).

r	Algorithms	*Performance*				
		Accuracy	*Round*	*Downstream Cost* (Kilobyte)	*Upstream Cost* (Kilobyte)	ϵ
0.5%	FL-BASIC	0.65	193	21402.03	107	N/A
	FL-CS	0.57	185	20514.9	102.56	N/A
	FL-TOP	**0.82**	200	**110.88**	**110.88**	N/A
	FL-BASIC-DP	0.59	200	22178.27	110.88	1
	FL-CS-DP	0.53	200	22178.27	110.88	1
	FL-TOP-DP	**0.81**	200	**110.88**	**110.88**	**1**
5%	FL-BASIC	0.78	196	21734.70	1086.73	N/A
	FL-CS	0.82	200	22178.27	1108.91	N/A
	FL-TOP	**0.84**	200	**1108.91**	**1108.91**	N/A
	FL-BASIC-DP	0.76	195	21623.81	1081.18	0.99
	FL-CS-DP	0.78	160	17742.61	887.13	0.94
	FL-TOP-DP	**0.81**	152	**842.77**	**842.77**	**0.92**
10%	FL-BASIC	0.81	196	21734.70	2173.47	N/A
	FL-CS	0.85	182	20182.22	2018.22	N/A
	FL-TOP	**0.85**	199	**2206.74**	**2206.74**	N/A
	FL-BASIC-DP	0.79	189	20958.46	2095.85	0.98
	FL-CS-DP	0.72	167	18518.85	1851.89	0.95
	FL-TOP-DP	**0.80**	157	**1740.99**	**1740.99**	**0.93**
100%	FL-STD	0.86	200	22178.27	22178.27	N/A
	FL-STD-DP	0.56	60	6653.48	6653.48	0.76

Table 1: Summary of results on Fashion-MNIST dataset.

r	Algorithms	*Performance*					
		Bal_Acc	*AUROC*	*Round*	*Downstream Cost* (Kilobyte)	*Upstream Cost* (Kilobyte)	ϵ
0.1%	FL-BASIC	0.51	0.51	99	11829.42	11.82	N/A
	FL-CS	0.53	0.55	100	11948.91	11.94	N/A
	FL-TOP	**0.69**	**0.76**	68	**8.12**	**8.12**	N/A
	FL-BASIC-DP	0.50	0.49	100	11948.91	11.94	1
	FL-CS-DP	0.51	0.51	99	11829.42	11.82	1
	FL-TOP-DP	**0.69**	**0.76**	85	**10.15**	**10.15**	**0.97**
5%	FL-BASIC	0.72	0.80	100	11948.91	597.45	N/A
	FL-CS	0.73	0.81	98	11709.93	585.5	N/A
	FL-TOP	**0.72**	**0.80**	95	**567.57**	**567.57**	N/A
	FL-BASIC-DP	0.69	0.76	100	11948.91	597.45	1
	FL-CS-DP	0.69	0.76	100	11948.91	597.45	1
	FL-TOP-DP	**0.68**	**0.75**	23	**137.41**	**137.41**	**0.79**
10%	FL-BASIC	0.74	0.81	100	11948.91	1194.89	N/A
	FL-CS	0.74	0.82	100	11948.91	1194.89	N/A
	FL-TOP	**0.74**	**0.82**	90	**1075.40**	**1075.40**	N/A
	FL-BASIC-DP	0.69	0.76	99	11829.42	1182.94	1
	FL-CS-DP	0.69	0.76	96	11470.95	1147.09	0.99
	FL-TOP-DP	**0.68**	**0.74**	23	**274.82**	**274.82**	**0.79**
100%	FL-STD	0.74	0.82	99	11829.42	11829.42	N/A
	FL-STD-DP	0.66	0.72	62	7408.32	7408.32	0.91

Table 2: Summary of results on Medical dataset.

These tables show that the proposed non-private scheme FL-TOP has similar accuracy than the standard scheme FL-STD but reduces the bandwidth cost significantly. For example, with the Fashion-MNIST dataset, the FL-TOP accuracy reaches 0.85 when the compression ratio $r = 10\%$. In comparison, the standard FL-STD scheme reaches an accuracy of 0.86% but consumes 10 times more bandwidth. Furthermore, although FL-CS reaches the same accuracy than FL-TOP and consumes slightly less bandwidth upstream (9% less), its required downstream bandwidth is about 10 times larger (See Table 1 for more details). The results on the medical dataset are quite similar. In fact, FL-TOP achieves its best balanced accuracy (0.74) and AUROC (0.82) when $r = 10\%$ while the FL-STD scheme obtains similar performance but required about 11 times more upsteam and downstream bandwidth cost. FL-CS achieves similarly accuracy at $r = 10\%$ as FL-TOP but its downstream required bandwidth is about 11 times larger (see Table 2 for more details).

The results also show that not only our privacy-preserving solution FL-TOP-DP provides strong privacy guarantee (with ϵ values smaller than 1) but that it outperforms the other schemes in term of accuracy and bandwidth, for both datasets. For example, with Fashion-MNIST, our scheme achieves an accuracy of 0.81 when $r = 0.5\%$ while the baseline scheme, FL-BASIC-DP, achieves an accuracy of 0.79 when $r = 10\%$ and requires 189 times more downstream bandwidth and 18 times more upstream bandwidth. With the medical dataset, FL-TOP-DP reaches the best balanced accuracy 0.69 and best AUROC 0.76 for a compression ratio of $r = 0.1\%$ while FL-BASIC-DP and FL-CS-DP achieves the same performance at $r = 5\%$. Note that FL-STD-DP performs very poorly as noise has to be added to the all weights of the model and the sensitivity is large (see Table 2).

5 RELATED WORK

Privacy of Federated Learning: The concept of Client-based Differential Privacy has been introduced in McMahan et al. [2018] and Geyer et al. [2017], where the goal is to hide any information that is specific to a single client's training data. These algorithms noise the contribution of a single client instead of a single record in the client's dataset.

The noise is added by the server, hence, unlike our solution, these works assume that the server is trusted.

Recently, Liu et al. [2020] also proposed to add noise only to the update of the Top-K model parameters a la local-DP. In local-DP, each client adds larger noise that what is necessary to guarantee DP for the *aggregated* model update without using secure aggregation. Therefore, the common model is less accurate than with our scheme. In addition, Liu et al. [2020] uses two epsilon budgets; one for selecting Top-K parameters per client, and the second for perturbing these selected Top-K parameters. By contrast, we select the Top-K parameters via public data without sacrificing any privacy budget. Finally, their solution is also less bandwidth efficient than ours: as the Top-K parameters differ for each client and at each round, the client cannot send only the Top-K parameters values because the server will not be able to identify which value corresponds to which Top-K parameter. For this reason, the client has to send a sparse vector with only Top-K perturbed values and all remaining parameters set to 0. Therefore, the quantization of the non-Top-K parameters is performed only during the upstream (from client to server) without compressing any downstream traffic. As opposed to this, in our solution, only the weights/updates of the Top-K parameters are transferred downstream/upstream.

Recently, Kerkouche et al. [2020] proposed to use Compressive sensing (CS) in the context of federated learning in order to compress model updates meanwhile providing client-level DP. Assuming that the model update is already sparse in the time domain, the noise is added to its largest Fourier coefficients in a distributed manner, and the noisy aggregate is reconstructed with standard optimization techniques. Likewise our solution, this work also uses secure aggregation by exploiting the linearity of CS. However, the reconstruction process can be slow for large models, and therefore our solution is more scalable. Moreover, it can only compress the upstream traffic.

Bandwidth Optimization in Federated Learning: Different quantization methods have been proposed to save the bandwidth and reduce the communication costs in federated learning. They can be divided into two main groups: unbiased and biased methods. The unbiased approximation techniques use probabilistic quantization schemes to compress the stochastic gradient and attempt to approximate the true gradient value as much as possible [Alistarh et al., 2016, Wen and al., 2017, Wang et al., 2018, Konecný et al., 2016]. However, biased approximations of the stochastic gradient can still guarantee convergence both in theory and practice [Bernstein et al., 2018, Lin et al., 2018, Seide et al., 2014]. SignSGD Bernstein et al. [2018] a quantization protocol allows to compress during downstream and upstream traffic but requires the use of all the clients at each round which is not realistic in the context of federated settings because each client is available only during few rounds Kairouz et al. [2019].

A different line of works exploit the sparsity of model updates to compress them. Amiri and Gündüz [2019a,b] proposed to use a compressive sensing for federated learning in order to compress model updates without privacy guarantees. However, they assume that all clients participate in each round (as they maintain an error accumulation vector at each client due to the compression scheme), but as discussed in Kairouz et al. [2019] this assumption is not always realistic. Sketching was adapted to federated learning for the purpose of compressing model updates in Ivkin et al. [2019] and Rothchild et al. [2020]. The authors proposed to use Count-Sketch from Charikar et al. [2002] to retrieve the largest weights in the update vector on the server side. However, it is unclear how these works can be extended with privacy guarantees. Moreover, unlike our technique, they do not compress downstream traffic.

Constraining the weights to have a specific distribution has already been studied. In Han et al. [2016], for example, the authors use pruning techniques to create a sparse model at the end of the training. After each SGD iteration, the authors zero-out all the weights with an absolute value smaller than a threshold. Iterating the process leads to a sparse model with only some absolute weight values larger than 0. Similarly, Courbariaux et al. [2016] aim to create a model with binary weights such that at the end of the training all the weights are close to 1 or -1. After each SGD update, the authors take the sign of the weights before the next update. After some iterations, the weight values become close to the interval limits -1 and 1.

In Frankle and Carbin [2018], a new hypothesis claims that there exists a sub-network which, if trained separately, can achieve similar performance as the complete network model which contains that. To find such a sub-network, one has to follow a simple iterative procedure: train the complete network, prune the smallest weights, and then reinitialize the remaining weights to their original values. These steps are repeated iteratively. This approach was extended to federated learning in Li et al. [2020].

6 CONCLUSION

This paper presents a novel privacy-preserving federated learning scheme that reduces bandwidth, latency and therefore power consumption. The proposed scheme is based on Differential Privacy and therefore provides theoretical privacy guarantees. Furthermore, it optimizes the model accuracy by constraining the model learning phase on few selected weights. We show experimentally, using a public dataset called Fashion-MNIST and a real world medical dataset of 1.2 million of US hospital patients, that it reduces the upstream and downstream bandwidth by up to 99.9% compared to standard federated learning, making it practical

for constrained and mobile devices.

Acknowledgements

This article was developed in the framework of the Grenoble Alpes Data Institute, supported by the French National Research Agency under the "Investissements d'avenir" program (ANR-15-IDEX-02). The research was supported by the NRDI fund of the Ministry of Innovation and Technology NRDI Office (Hungary), and also within the framework of the Artificial Intelligence National Laboratory Program. This project has received support from the ANR project ANR-16-CE25-0010.

References

Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *ACM CCS*, 2016.

Gergely Ács and Claude Castelluccia. I have a dream! (differentially private smart metering). In *IH*, 2011.

Dan Alistarh, Jerry Li, Ryota Tomioka, and Milan Vojnovic. QSGD: randomized quantization for communication-optimal stochastic gradient descent. 2016.

Mohammad Mohammadi Amiri and Deniz Gündüz. Machine learning at the wireless edge: Distributed stochastic gradient descent over-the-air. 2019a.

Mohammad Mohammadi Amiri and Deniz Gündüz. Federated learning over wireless fading channels. 2019b.

Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Anima Anandkumar. signsgd: compressed optimisation for non-convex problems. 2018.

Keith Bonawitz et al. Practical secure aggregation for federated learning on user-held data. 2016.

Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *International Colloquium on Automata, Languages, and Programming*, pages 693–703. Springer, 2002.

Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1, 2016.

Cynthia Dwork and Aaron Roth. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4), 2014.

Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. RAPPOR: randomized aggregatable privacy-preserving ordinal response. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, pages 1054–1067. ACM, 2014. doi: 10.1145/2660267.2660348. URL https://doi.org/10.1145/2660267.2660348.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Training pruned neural networks. 2018.

Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients – how easy is it to break privacy in federated learning?, 2020.

Robin C. Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. 2017.

Song Han, Jeff Pool, Sharan Narang, Huizi Mao, Enhao Gong, Shijian Tang, Erich Elsen, Peter Vajda, Manohar Paluri, John Tran, Bryan Catanzaro, and William J. Dally. Dsd: Dense-sparse-dense training for deep neural networks, 2016.

Nikita Ivkin, Daniel Rothchild, Enayat Ullah, Ion Stoica, Raman Arora, et al. Communication-efficient distributed sgd with sketching. In *Advances in Neural Information Processing Systems*, pages 13144–13154, 2019.

Peter Kairouz et al. Advances and open problems in federated learning. 2019.

Raouf Kerkouche, Gergely Ács, Claude Castelluccia, and Pierre Genevès. Compression boosts differentially private federated learning, 2020. To appear in EuroS&P 2021.

Jakub Konecný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. 2016.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/.

Ang Li, Jingwei Sun, Binghui Wang, Lin Duan, Sicheng Li, Yiran Chen, and Hai Li. Lotteryfl: Personalized and communication-efficient federated learning with lottery ticket hypothesis on non-iid datasets, 2020.

Yujun Lin, Song Han, Huizi Mao, Yu Wang, and Bill Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *ICLR*, 2018.

Ruixuan Liu, Yang Cao, Masatoshi Yoshikawa, and Hong Chen. Fedsel: Federated sgd under local differential privacy with top-k dimension selection. *Lecture Notes in Computer Science*, 2020.

H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, 2016.

H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. In *International Conference on Learning Representations*, 2018.

Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Inference attacks against collaborative learning. 2018.

Ilya Mironov, Kunal Talwar, and Li Zhang. Rényi differential privacy of the sampled gaussian mechanism. 2019.

Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *IEEE Symposium on Security and Privacy*, 2019.

Daniel Rothchild, Ashwinee Panda, Enayat Ullah, Nikita Ivkin, Ion Stoica, Vladimir Braverman, Joseph Gonzalez, and Raman Arora. Fetchsgd: Communication-efficient federated learning with sketching, 2020.

Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns. In *INTERSPEECH*, 2014.

Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *CCS*, 2015.

Hongyi Wang et al. Atomo: Communication-efficient learning via atomic sparsification. In *NeurIPS*, 2018.

Wei Wen and al. Terngrad: Ternary gradients to reduce communication in distributed deep learning. 2017.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. 2017.

Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. idlg: Improved deep leakage from gradients. 2020.

Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *NeurIPS 2019*, 2019.

Yuqing Zhu, Xiang Yu, Yi-Hsuan Tsai, Francesco Pittaluga, Masoud Faraki, Manmohan chandraker, and Yu-Xiang Wang. Voting-based approaches for differentially private federated learning, 2020.