Privacy Preserving Distributed Deep Learning and its Application in Credit Card Fraud Detection

Yang Wang*, Stephen Adams*, Peter Beling*, Steven Greenspan†,
Sridhar Rajagopalan†, Steven Boker‡, and Donald Brown*
*Department of Systems and Information Engineering, University of Virginia
Charlottesville, VA, 22903
†CA Technologies
200 Princeton S. Corp Center, Ewing, NJ 08628
‡Department of Psychology, University of Virginia
Charlottesville, VA, 22903

***Abstract*—Preserving privacy in machine learning on multi-party data is of importance to many domains. Existing solutions suffer from several critical limitations, such as significantly reduced utility after enforcing differential privacy, excessive communications burden between information fusion center and local participants who contribute data, etc. These severely limit their practical adoption. In this paper, we propose and implement a new distributed deep learning framework that addresses these shortcomings and preserves privacy in a more accurate and efficient way than prior methods. During the stochastic gradient descent process in training a deep neural network, we focus on the parameters with large gradient values to save privacy budget consumption, and adopt a generalization of the Report-Noisy-Max algorithm in differential privacy to select and release these gradients in order to prevent indirect privacy leakage. Inspired by the recent novel work of [1], we also limit the shared gradient for each parameter to be one of three floating numbers $\{-B, 0, B\}$, where B is the bound for each gradient. This method can significantly reduce the communication burden without severely affecting accuracy. Furthermore, we evaluate the performance of our system on a real-world credit card fraud dataset consisting of millions of transactions.**

***Index Terms*—differential privacy, deep learning, distributed machine learning, credit card fraud detection**

I. Introduction

Detecting fraudulent credit card transactions is one of the biggest challenges in the banking industry. Modern technology, especially innovation in machine learning, has been applied to both analyze the spending patterns of customers and to block transactions that are irregular or possibly fraudulent [2]–[5]. Among various machine learning approaches, neural network models have demonstrated exceptional performance, provided an abundant supply of labeled training data is available [6], [7]. Financial transaction information, however, is considered sensitive in both its relationship to customer privacy and its importance as a source of proprietary advantage to banks. Because of these privacy issues, neural network models for fraud detection are typically trained using only the in-house data collected by each bank individually. The increasing concern over data privacy imposes restrictions and barriers to data sharing and make it difficult to coordinate large-scale collaborative studies. Privacy preserving multi-party machine learning offers the promise that it can take advantage of data sets from different banks and thereby construct a model superior to stand-alone efforts.

Consider the example where the transactions in different banks or credit card companies focus on different merchant categories (e.g. retail stores, wholesale trade, transportation). The spending patterns of customers differ vastly across merchant categories and stand-alone efforts to build fraud detection model in each bank fail to generalize well to unseen transactions due to lack of information in other merchant categories. To resolve this impasse, it is crucial to design a multi-party machine learning mechanism to facilitate the collaboration among different banks without violating customer privacy or leaking business secret. It should be noted that similar opportunities are present in other fields as well [8]. In medical or psychological studies, for example, central aggregation of data sets may not be possible or permitted due to logistic complexities or privacy concerns associated with patient records or sensitive human-subject data.

This paper aims to provide a privacy preserving method for training deep neural networks from multiple sources of data. The performance of the proposed method is evaluated using a real-world credit card fraud detection dataset. Our work bridges the gap between collaborative machine learning and privacy-preserving machine learning, with the aim of providing a framework that can be the basis for banks and other entities to share their data without revealing sensitive information.

There have been various attempts to address the privacy challenge, such as anonymization and encryption [9]–[11]. However, some of the existing solutions either have high communication and computation burden or do not have mathematically rigorous privacy guarantees [12], [13]. Indeed, many of the previous work along these lines have been proven to not be private at all [14], [15]. Here, we follow a prevalent theoretical framework for differential privacy [16] and propose a differentially private and effective method to train a deep neural network. There are numerous efforts focused on training differentially private machine learning models from multi-party data sets. One line of work is to develop algorithms to

combine classifiers from different participants. Pathak, Rane, and Raj [17] leverage cryptography to securely average over local classifiers and release a perturbed version of the averaged model. This work lacks theoretical justification regarding accuracy optimality of the averaged model (as critiqued by [18]), and performs poorly when irregular or imbalanced data are present across different organizations. A recent paper [19] proposes an ensemble learning (of majority voting) and (pseudo)labeling solution to make the previous solution less restricted to averaging. It designates a *trusted authority* to compose an ensemble model of local models and use this model to label privacy-free (public) auxiliary data. The newly-labeled data are then used to train a global differentially private classifier, which can be released safely. The difficulty with this approach is the assumption that a trusted authority exists; the primary motivation for multi-party learning derives from the impossibility of establishing such an entity in practice.

Another line of work attempts to improve over [20] by proposing differentially private versions of iterative numerical optimizers, such as private (stochastic) gradient descent. These solutions often work by adding noise to the gradients in each iteration. However, it is possible that the model performance will be severely affected for practical deployment in multi-party learning. Moreover, in a large model like a deep neural network, there are often hundreds of thousands parameters to train and the excessive interactions/iterations results in fairly quick privacy budget consumption.

In summary, our principal contributions are:

- We provide end-to-end differential privacy guarantee on distributed deep learning, which improves recent proposals in terms of utility and privacy budget consumption.
- We propose a novel differentially private algorithm for choosing the most important gradients in each iteration and prove its privacy properties rigorously.
- We adapt the newly proposed idea of Terngrad to implement an efficient learning process with less communication bandwidth requirement.
- We evaluate our method on a real-world fraud detection data set with multi-million transactions and achieve performances comparable to previous work even under privacy constraints.

The next section reviews basic concepts and theorems in differential privacy. Section III and Section IV present our method and give a theoretical analysis of its privacy-preserving property. Section V describes the dataset and the experimental results. Finally, in Section VI, we conclude our work. Proofs are given in the Appendix.

II. Preliminaries

Differential privacy [21] is a mathematically rigorous notion of privacy. More formally:

Definition 2.1 (Differential Privacy [22]): A randomized algorithm M (with output space Ω and well-defined probability density P) is (ϵ, δ)-differentially private if for all adjacent data sets D, D' that differ in a single record and for all measurable sets $\omega \in \Omega$:

$$Pr[M(D) \in \omega] \leq e^{\epsilon} Pr[M(D') \in \omega] + \delta.$$

Differential privacy essentially implies that even when a strong adversary knows the whole private dataset D except for the target record, he or she still cannot infer much information about the existence of the target in D because the output of the randomized algorithm is almost unchanged. The parameter ϵ is also defined as the privacy budget which quantifies the level of privacy guarantee of the algorithm. When we think about ϵ, we should understand it as a "budget" rather than a purely statistical upper bound of the probability ratio in definition 2.1. A differentially private algorithm with smaller ϵ means that it is more private. Every time the private dataset of a participant is queried by a ϵ-differentially private algorithm (e.g. gradient calculation), the amount of the privacy budget ϵ will be consumed. When the private dataset is queried by a composition of algorithms, the consumed privacy budget will sum up following the composition theorems for differential privacy [23]. In practice, each participant sets a total privacy budget based on the desired level of privacy guarantee. When the consumed privacy budget exceeds the total budget, algorithms or queries can no longer be applied to the dataset. The other privacy parameter δ quantifies the probability of the algorithm failing to satisfy differential privacy and is usually set to a very small number or zero.

A popular way to achieve differential privacy in machine learning algorithms is output perturbation, which works by adding noise to the output. The amount of noise is carefully calibrated to the sensitivity of the output and ϵ. The sensitivity measures the maximum change of the algorithm when a single record in the input data is changed.

Definition 2.2 (Sensitivity [22]): The l_2 sensitivity of function f is defined as:

$$S(f) = \max_{D,D'} \|f(D) - f(D')\|_2 \ ,$$

where $\|.\|_2$ denotes the l_2 or Euclidean norm and D, D' are two neighboring datasets differeing only in one data point.

The Laplace Mechanism is one of the standard output perturbation methods in differential privacy. It has been shown to preserve $(\epsilon, 0)$-differential privacy [22].

Definition 2.3 (The Laplace Mechanism [22]): Given any function $f : D \to R^d$, the Laplace mechanism M is defined as:

$$M(D, f, \epsilon) = f(D) + (n_1, ..., n_d)$$

where n_i are i.i.d. random variables drawn from the Laplacian distribution $\mathbf{Lap}(S(f)/\epsilon)$.

With the basic techniques for designing a differentially private algorithm, it is also important to investigate how much privacy budget is consumed after composition of various differentially private algorithms. For example, when training a neural network, it is necessary to iterate numerous times over the dataset in order to optimize the parameters due to the non-convexity of the objective function. Reducing the

consumption of the privacy budget while achieving satisfactory model performance remains an interesting and challenging question. The following theorems are the basic and advanced composition theorems in differential privacy and based on these theorems we will present our approach to controlling privacy budget consumption in the next section.

Theorem 2.4 (Basic composition theorem [22]): Let M_i be an (ϵ_i, δ_i)-differentially private algorithm $(i = 1, 2, ..., K)$. Then the composition of all the K algorithms $M_{[K]}(D) = (M_1(D), ..., M_K(D))$ is $(\sum_i \epsilon_i, \sum_i \delta_i)$-differentially private.

Theorem 2.5 (Advanced composition theorem [22]): For all $\epsilon, \delta, \delta' \geq 0$, the class of (ϵ, δ)-differentially private mechanisms satisfies $(\epsilon', k\delta + \delta')$-differential privacy under k-fold adaptive composition for:

$$\epsilon' = \sqrt{2k \ln(1/\delta')}\epsilon + k\epsilon(e^{\epsilon} - 1).$$

III. Methods

In this section, we present our privacy-preserving distributed deep learning system in detail, which enables multiple participants to jointly train a neural network model without sharing raw data. There are in total K participants (banks), each owning a private dataset $D^k = \{\mathbf{X}_i^k, y_i^k\}|_{i=1}^{n_k}$ $(k = 1, 2, 3, ..., K)$, where $\mathbf{X}_i^k$ is the feature vector, y_i^k is the corresponding label and n_k is the size of the dataset associated with participant k. In addition, there is a parameter server, which oversees information integration and dissemination. All the participants will communicate directly with the server (fetching parameters, uploading gradients, etc). The server is responsible for updating the parameters iteratively and make them available to all participants. Here we assume both the participants and the server are honest but curious. They will follow the protocols honestly, but meanwhile try to learn as much information as possible about the other participants from their views of the protocols. It can be potentially privacy leaking as shown in [24]. Therefore, all the information coming out of a participant has to be differentially private in order to protect its private dataset. The structure and basic workflow of the distributed learning framework is illustrated in figure 1.

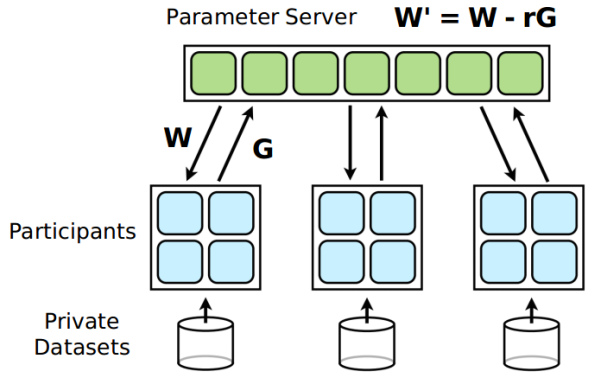

Fig. 1. Diagram of the distributed deep learning system. W represents the parameters and G represents the gradient information.

In our privacy-preserving multi-banks credit card fraud detection system, we are aiming to protect the privacy of bank's customers, or more exactly, whether a certain transaction happened in a certain bank, by enforcing differential privacy. Before jointly training the models, all participants will first agree on a common model (the architecture of the neural network, features, activation function in each hidden layer, loss functions etc). The parameter server will initialize the parameters randomly and pass them to the first participant. The first participant trains the model on its own private dataset and uploads the differentially private gradients to the server, who updates the parameters accordingly. Then the server will contact the next participant and pass the new parameters to it for more training, and so on. The whole training process will go on for T iterations. During each iteration, all the participants will be contacted once in a round-robin manner. The privacy budget consumption for each participant can be carefully calculated and maintained. Following the setting of most existing works [25], [26], each participant will fix its total privacy budget and number of iterations ahead of time and avoid adaptive choice of privacy parameters.

During each iteration, after downloading the current parameters from the server, the participant will sample a small subset from its own private dataset and compute the average gradient on this subset. To deal with exploding gradients problem, the gradient will first be clipped element-wise if it exceeds in absolute value a fix bound B , which is a common trick in deep learning [27], [28]. Then we apply the differentially private Report-Noisy-Top-N (**RNTN**) algorithm (Algorithm 3) repeatedly in order to pick the parameters with large absolute gradient values. By focusing on these "effective" gradients, we can save privacy budget because the privacy budget spent is linearly related to the number of gradients uploaded to the server. A similar idea was adopted in the work of Shokri and Shmatikov [25], which used a sparse vector technique for this purpose, but the privacy and accuracy analysis was incomplete and unclear. Moreover, instead of uploading the noisy gradients to the server, we upload the "ternary gradient" $\{-B, 0, B\}$ based on the sign of the selected noisy gradients in order to reduce the communication burden. This idea is proposed by the recent work of Wen *et al.* [1].

The details of the model training process are described in Algorithms 1 through 3. Algorithm 1 is the overall parameter update process controlled by the parameter server. Algorithm 2 is the noisy gradients calculation and uploading process going on in a participant during each iteration. Algorithm 3 describes how to pick the top noisy gradients in a differentially private way.

IV. Privacy Analysis

In this section, we analyze the privacy-preserving property of the proposed algorithms. First we will demonstrate that mini-batch stochastic gradient descent method in each iteration will greatly save the privacy budget consumption.

Algorithm 1 Server Side

Require: neural network model M (neural network structure, activation function, loss function $L(\mathbf{W},\ \mathbf{X},\ y)$, features schema, etc), total iteration T, number of participants K, learning rate at iteration t $\{r_t\}$

Ensure: final differentially private parameter $\mathbf{W} \in R^d$

1: Initialize the parameter $\mathbf{W}$ (by default initialized to 0) and disseminate the model M to all the participants
2: **for** $t = 1$ to T **do**
3: **for** $k = 1$ to K **do**
4: Communicate with the k-th participant and pass the current parameter $\mathbf{W}$ to it
5: Receive the differentially private ternary gradient vector $\tilde{\mathbf{G}} = \mathbf{DPSTGD}()$ from the participant k.
6: Update the current parameters $\mathbf{W} := \mathbf{W} - r_t\tilde{\mathbf{G}}$
7: **end for**
8: **end for**
9: Return $\mathbf{W}$

Lemma 4.1: Let D be a dataset and M be a ϵ-differentially private algorithm on the domain of D. Define the algorithm M_q as follows:

1) Sample a random subset D_q from D with sampling probability q
2) Run M on D_q and return the results

Then M_q is a $\ln(1+(e^\epsilon-1)q)$-differentially private algorithm.

See Appendix A for proof.

Note that $\ln(1 + (e^\epsilon - 1)q) < \epsilon$ when $0 < q < 1$. Lemma 4.1 confirms that by sampling from the whole dataset for the gradient calculation, which is a common method in stochastic gradient descent and deep neural network, we can save on spending the privacy budget. The savings are illustrated in Figure 2. When ϵ is small, the saving is almost linear with q. However, when ϵ is large, the sampling does not have a significant impact on the privacy budget saving since the saving vanishes quickly with respect to q. Luckily, in our situation, the privacy budget ϵ for each iteration is small because we are expecting many iterations and many parameters. Therefore, mini-batch sampling is indeed a great way to save on the privacy budget.

The following theorem provides a rigorous proof of the privacy preserving property of our proposed method.

Theorem 4.2: Algorithm 2 is $(2n\epsilon_g, 0)$-differentially private, where

$$\epsilon_g = \ln(1 + (e^\epsilon - 1)q).$$

Alternatively, for any cryptographically small privacy parameter δ (e.g. 2^{-30}), Algorithm 2 is $(\sqrt{4n\ln(1/\delta)}\epsilon_g + 2n\epsilon_g(e^{\epsilon_g} - 1), \delta)$-differentially private.

See Appendix B for proof.

Next, we will present the utility analysis of the **RNTN** algorithm (Algorithm 3) in selecting the top largest absolute gradients. Since it is a generalization of **Report-Noisy-Max** [22], we will focus on bounding the probability ratio of selecting the largest value and second largest value in the **Report-Noisy-Max** algorithm.

Algorithm 2 Participant Side: differentially private Stochastic Ternary Gradient Descent (**DPSTGD**)

Require: parameters $\mathbf{W} \in R^d$, loss function $L(\mathbf{W},\ \mathbf{X},\ y)$, privacy budget ϵ, number of gradients to upload n, mini-batch sampling ratio q, private dataset D, gradient bound $B > 0$

Ensure: ternary gradient vector $\tilde{\mathbf{G}} \in \{-B, 0, B\}^d$

1: Randomly sample a subset D_q from the private dataset D with sampling probability q.
2: For each point $(\mathbf{X}_i,\ y_i)$ in the subset D_q $(i = 1, 2, ..., |D_q|)$:
calculate gradient:

$$\mathbf{g}_i = \nabla L(\mathbf{W},\ \mathbf{X}_i,\ y_i)$$

clip gradient:

$$\hat{\mathbf{g}}_i = \mathbf{g}_i / \max(1, \frac{|\mathbf{g}_i|}{B})$$

3: Take the average of the clipped gradients of all the points in D_q

$$\mathbf{G} := \frac{1}{|D_q|}\sum_i \hat{\mathbf{g}}_i$$

4: Calculate sensitivity λ of the absolute value of each gradient j $(j = 1, 2, ..., d)$

$$\begin{aligned}\lambda &= \max_{D,D'}(|\mathbf{G}^j(D)| - |\mathbf{G}^j(D')|) \\ &= \max_{D,D'} \frac{1}{|D_q|}(|\sum_i \hat{\mathbf{g}}_i^j(D)| - |\sum_i \hat{\mathbf{g}}_i^j(D')|) \\ &= \frac{2B}{|D_q|}\end{aligned}$$

5: Run **Report-Noisy-Top-N()** with privacy budget ϵ and sensitivity λ to pick the index set $\mathbf{I}$ of the n gradients with the largest noisy absolute values.
6: For each j in the index set $\mathbf{I}$, set $\tilde{\mathbf{G}}^j = \text{sign}\left(\mathbf{G}^j + \mathbf{Lap}\left(\frac{\lambda}{\epsilon}\right)\right) \times B$; otherwise, set $\tilde{\mathbf{G}}^j = 0$.
7: Return $\tilde{\mathbf{G}}$.

*Theorem 4.3 (Accuracy analysis of **Report-Noisy-Max**):* Let ϵ be the privacy parameter in the **Report-Noisy-Max** algorithm and λ be the sensitivity of absolute value of gradient. The probability of selecting the gradient with largest absolute value is at least $\exp\left(\frac{\Delta\epsilon}{\lambda}\right) / \left(\frac{\Delta\epsilon}{4\lambda} + \frac{1}{2}\right) - 1$ larger than that of selecting gradient with the second largest absolute value, where Δ is the difference between the largest absolute gradient and the second largest absolute gradient.

See Appendix C for proof.

The theorem above gives utility guarantee in the sense that small gradients are exponentially less likely selected

Algorithm 3 Report-Noisy-Top-N (**RNTN**)

Require: privacy budget ϵ, gradient functions g_i $(i = 1, 2, ..., d)$ each with sensitivity at most λ, number of gradients to select N, dataset D

Ensure: index and noisy values of the top N gradients with largest noisy values

1: Compute the noisy gradients of D: $\hat{g}_i(D) = g_i(D) + \nu_i$, where $\nu_i \sim \mathbf{Lap}(\lambda/\epsilon)$.

2: Return the top N indices with the largest noisy values, and also their noisy values $\hat{g}_i(D)$ $(i \in I)$.

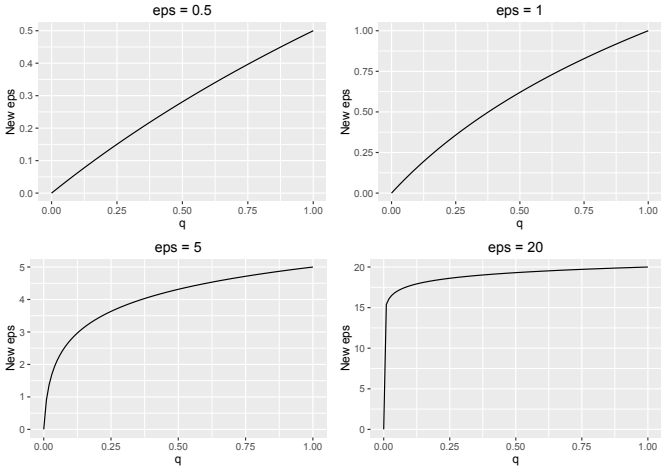

Fig. 2. Visual illustration of Lemma 1: "eps" is the privacy budget consumed without mini-batch sampling and "new eps" is the privacy budget consumed under varying mini-batch sampling ratio.

by **Report-Noisy-Max** algorithm after adding noise to the gradients.

V. Experiments

The main goal of the experiment is to implement our method and evaluate its utility-privacy trade-off on a real-world credit card fraud dataset. The objective of the dataset is to detect fraudulent transactions given information about the transaction. Imposing privacy requirements will generally degrade the model performance due to the random noise added to the gradients. We build and train neural networks on the pre-processed dataset under varying settings of privacy parameters, mini-batch ratio and ratio of uploaded gradients. In particular, we are interested in how the performance of the neural network varies with the privacy budget, which determines the amount of noise. The results demonstrate that the proposed method achieves performance close to the non-private baseline even under relatively strict privacy requirement.

A. Dataset Preprocessing

In the experiment, a real-world credit card transaction data set contributed by a US bank is studied. All the 78 million transactions from over 2 million accounts happened in the first eight month of 2013 and are labeled as either non-fraudulent or fraudulent. There are 69 categorical and numeric features in the original dataset, including transaction amount, transaction date and time, merchant code, account-level information, card present or not, distance to merchant, etc. In addition, we derived new features reflecting the spending pattern of customers with the help of domain expertise, such as amount difference and distance difference from last transaction, time duration since last transaction, velocity variables , standard deviation of transaction amount for each account over time, etc. We apply effect coding to all the categorical variables and standardize all numerical variables. There are in total 264 features in the processed data set.

The percentage of fraudulent transaction in the entire data set is 0.136%. Since it is extremely unbalanced, we decide to under-sample the non-fraudulent transactions at the account-level. To be more specific, we kept all the accounts with fraudulent transactions and randomly sample the same amount of accounts from the remaining accounts without fraudulent transactions. All the fraudulent transactions and a random 20% of all the non-fraudulent transactions from these accounts are used as our sample for building and testing the neural network model. After pre-processing, there are about 0.7 million transactions with a fraud rate of 14% in the sample.

B. Results

The model we are trying to build is a generic neural network with 2 hidden layers of neurons. The number of neurons in each hidden layer are 50 and 20 respectively. Therefore, the total number of weights and bias parameters in the neural network is 14,312. To simulate the multi-participant situation, the accounts are divided evenly into each participant and also a test set.

First, we present the amount of the privacy budget consumed at each iteration and how the privacy budget is affected by the number of uploaded gradients n and the mini-batch sampling ratio q. The results are displayed in table I. We can see that without mini-batch sampling and top gradients selection, the privacy budget consumed at each iteration will explode, which makes the differentially private method impractical. Sampling a mini-batch and uploading a proportion of all the gradients will save a lot privacy budget without hurting the model performance (see the experiments below). Moreover, if we can tolerate a tiny probability of the algorithm failing to preserve privacy (e.g. $\delta = 2^{-30}$), the advanced composition theorem will greatly save the privacy budget compared to the basic composition theorem.

To illustrate the utility-privacy trade-off, we compare the AUC of the model on the test set of our approach to a non-private neural network baseline under different values of ϵ. In addition, we also test the impact of the number of participants in the systems, the mini-batch sampling ratio, and the number of uploaded gradients. The whole analysis was repeated 10 times with different seeds for random number generator. The results are very similar for different trials, for sake of clarity only a single result is presented.

Figure 3 shows the AUC on the test set under different privacy budgets in each iteration for 5 participants. We set

TABLE I

PRIVACY BUDGET CONSUMPTION AT EACH ITERATION. ϵ IS THE PRIVACY PARAMETER IN ALGORITHM 2, q IS THE MINI-BATCH SAMPLING RATIO, AND n IS THE NUMBER OF UPLOADED GRADIENTS. ϵ_{iter}^{basic} IS THE PRIVACY BUDGET CONSUMED BY BASIC COMPOSITION THEOREM; ϵ_{iter}^{adv} IS THE PRIVACY BUDGET CONSUMED BY ADVANCED COMPOSITION THEOREM WHEN $\delta = 2^{-30}$

ϵ	q	n	ϵ_{iter}^{basic}	ϵ_{iter}^{adv}
0.1	0.01	1431	3.01	0.37
0.1	0.01	2862	6.02	0.52
0.1	0.05	1431	15.01	1.88
0.1	1	14312	2862.4	410.1
0.5	0.01	1431	18.50	2.35
0.5	0.01	2862	37.01	3.39
0.5	0.05	1431	91.35	13.97
0.5	1	14312	14312.4	9830.1

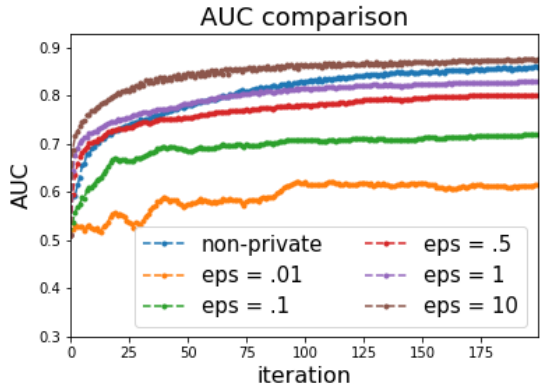

Fig. 3. AUC comparision under different privacy budget (5 participants, mini-batch ratio = 0.01, ratio of uploaded gradient = 10%)

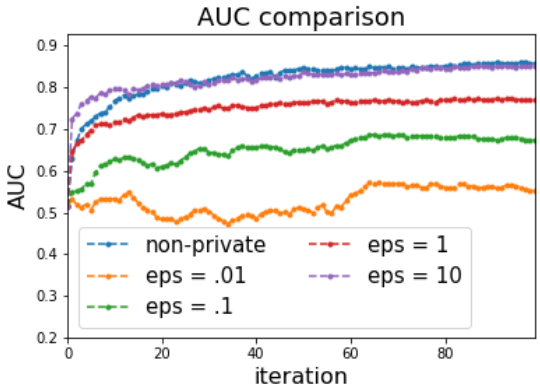

Fig. 4. AUC comparision under different privacy budget (30 participants, mini-batch ratio = 0.01, ratio of uploaded gradient = 10%)

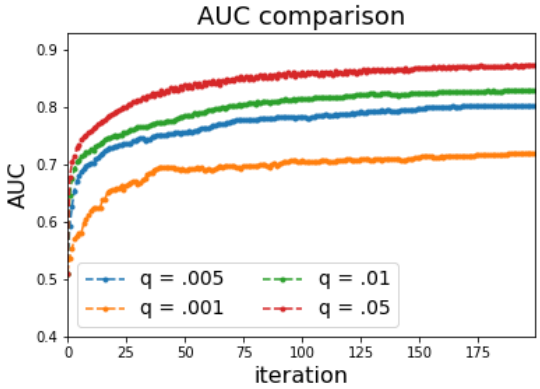

Fig. 5. AUC comparision under mini-batch ratio (5 participants, $\epsilon = 1$, ratio of uploaded gradient = 10%)

the mini-batch ratio to be 0.01 and for our **DPSTGD** algorithm, only 10% of all the gradients are uploaded to the sever in each iteration. As expected, a smaller privacy budget ($\epsilon = .01, .1$) gives relatively weaker performance. For an intermediate differential privacy guarantee ($\epsilon = .5, 1$), the performance is comparable to the non-private baseline. It is also worth noting that when the privacy budget is large ($\epsilon = 10$), the performance of our method even beats the non-private baseline. This experiment is a proof-of-concept that privacy preservation can give satisfactory performance even under an intermediate differential privacy guarantee on real-world data set. Moreover, adding a small amount of noise and focusing on the top gradients will not only reduce the communication burden but also help learning the model faster and improving the accuracy. It is also consistent with the phenomenon studied in the paper [29]. Our understanding is that adding noise to gradients mostly likely helps the optimizer jump out of local minimum for a non-convex problem like deep neural network whereas exact gradient descent method will get the optimizer stuck.

Figure 4 presents the results when there are 30 participants. The trend of AUC curves is similar to that of Figure 3.

Lastly, we look at the impact of mini-batch ratios and ratio of uploaded gradients with $\epsilon = 1$. As expected, Figure 5 and 6 show that when the mini-batch ratio and the ratio of uploaded gradients are increased, the performance are also improved. Note that even if only 30% of all the noisy ternary gradients are uploaded to the server, the model performance is superior to that of the non-private baseline. It further verifies the impression that for a deep neural network, only a proportion of

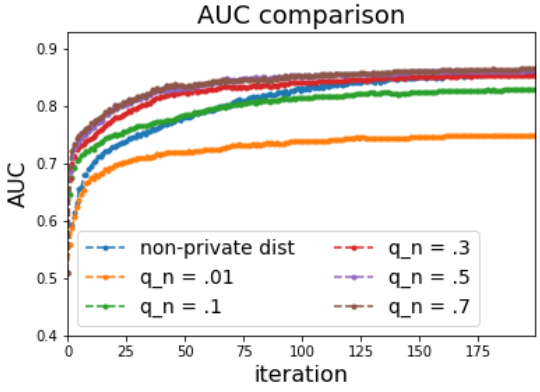

Fig. 6. AUC comparision under mini-batch ratio (5 participants, $\epsilon = 1$, mini-batch ratio = 1%)

the gradients are large enough to change the parameter and the value of loss function, and our **Report-Noisy-Top-N** algorithm is effective in selecting these gradients.

VI. Conclusion

In this paper, we propose a privacy-preserving method to train deep neural networks in a distributed setting and evaluate its utility-privacy trade-off on a real-world credit card fraud detection dataset. In order to save privacy budget consumption, we sample a mini-batch from the private dataset from each participant and focus on the large gradients. In addition, we upload ternary gradients instead of the exact gradients to reduce communication burden. Our privacy-preserving method achieves model performance (measured by AUC on test set) comparable to the non-private baseline. It provides a practical solution to the multi-party privacy preserving deep learning, which is especially beneficial to financial institutions who are willing to jointly learning machine learning models, but are prohibited by privacy restriction.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. CNS: 1650512. This work was conducted in the NSF UICRC Center of Visual and Decision Dynamics, through the sponsorship and guidance of CA Technologies. We thank (folks on the team, who choose not be authors) for their thoughts on this paper and the overall project.

Appendix A
Proof of the Lemma 4.1

Proof: Let D and D' be two neighboring datasets such that $D' = D \cup \{x\}$. For any event O in the output space of the algorithm M, we consider the following two cases:

1) x is not in D'_q (probability: $1-q$)
2) x is in D'_q (probability: q)

When x is not in D'_q, the output distribution of running M_q on D is the same as that of running M_q on D'. Therefore,

$$Pr(M_q(D') \in O) = Pr(M_q(D) \in O).$$

When x is in D'_q, by the definition of $(\epsilon, 0)$-differential mechanism, we have

$$Pr(M_q(D') \in O) \leq e^{\epsilon} Pr(M_q(D) \in O).$$

Taking the probability of each case into account, we have

$$\begin{aligned}
&Pr(M_q(D') \in O) = \\
&(1-q) \times Pr(M_q(D') \in O | x \notin D'_q) + \\
&q \times Pr(M_q(D') \in O | x \in D'_q) \\
&\leq (1-q) Pr(M_q(D) \in O) + q e^{\epsilon} Pr(M_q(D) \in O) \\
&= (1 + (e^{\epsilon} - 1)q) Pr(M_q(D) \in O).
\end{aligned}$$

Therefore, by the definition of differential privacy, M_q is $(\ln(1 + (e^{\epsilon} - 1)q), 0)$-differentially private on the domain of D. ∎

Appendix B
Proof of Theorem 4.2

Proof: First, we claim that algorithm **Report-Noisy-Top-N** is $2n\epsilon$-differentially private. This is a generalization of the **Report-Noisy-Max** algorithm in [22] and we skip the proof here.

By Lemma 4.1, if only a mini-batch q of the private data set is sampled during each iteration, then the privacy budget spent by **Report-Noisy-Top-N** per uploaded gradient is

$$\epsilon_g = \ln(1 + (e^{\epsilon} - 1)q).$$

From basic composition theorem 2.4, the total privacy budget spent on uploading N noisy gradients during each iteration is $2n\epsilon_g$.

Alternatively, from advanced composition theorem 2.5, for any tiny privacy parameter δ, algorithm 2 is $(\sqrt{4n \ln(1/\delta)}\epsilon_g + 2n\epsilon_g(e^{\epsilon_g} - 1), \delta)$-differentially private. ∎

Appendix C
Proof of Theorem 4.3

Proof: First, given N absolute values of gradients $|g_1|$, $|g_2|$,..., $|g_N|$, we derive the probability P_i of selecting an arbitrary i-th gradient $|g_i|$ by **Report-Noisy-Max** algorithm.

$$\begin{aligned}
P_i &= \prod_{j=1,..,N, j \neq i} Pr(|g_i| + n_i > |g_j| + n_j) \\
&= \prod_{j=1,..,N, j \neq i} Pr(n_i > n_j + \Delta_{j,i}),
\end{aligned}$$

where n_i, $n_j \sim \text{Lap}(\lambda/\epsilon)$ and $\Delta_{j,i} = |g_j| - |g_i|$. Then we have

$$\begin{aligned}
&Pr(n_i > n_j + \Delta_{j,i}) \\
&= \int_{-\infty}^{\infty} Pr(n_i > n_j + \Delta_{j,i} | n_i = t) Pr(n_i = t)\, \mathrm{d}t \\
&= \int_{-\infty}^{\infty} Pr(n_j < t - \Delta_{j,i}) Pr(n_i = t)\, \mathrm{d}t \\
&= \frac{\epsilon}{4\lambda} \int_{-\infty}^{\Delta_{j,i}} \exp(\frac{\epsilon t - \epsilon|t| - \Delta_{j,i}\epsilon}{\lambda})\, \mathrm{d}t \\
&\quad + \frac{\epsilon}{2\lambda} \int_{\Delta_{j,i}}^{\infty} \exp(-\frac{\epsilon|t|}{\lambda}) \left[1 - \frac{1}{2}\exp(\frac{\Delta_{j,i}\epsilon - \epsilon t}{\lambda})\right] \mathrm{d}t.
\end{aligned} \tag{1}$$

We consider the following two cases:

1) When $\Delta_{j,i} > 0$, equation 1 becomes

$$\begin{aligned} &Pr(n_i > n_j + \Delta_{j,i}) \\ &= \frac{\epsilon}{4\lambda}\exp(-\frac{\Delta_{j,i}\epsilon}{\lambda})\int_{-\infty}^{0}\exp(\frac{2\epsilon t}{\lambda})\,\mathrm{d}t \\ &\quad + \frac{\epsilon}{4\lambda}\exp(-\frac{\Delta_{j,i}\epsilon}{\lambda})\int_{0}^{\Delta_{j,i}}\mathrm{d}t \\ &\quad + \frac{\epsilon}{2\lambda}\int_{\Delta_{j,i}}^{\infty}\exp(-\frac{\epsilon t}{\lambda})\,\mathrm{d}t \\ &\quad - \frac{\epsilon}{4\lambda}\exp(\frac{\Delta_{j,i}\epsilon}{\lambda})\int_{\Delta_{j,i}}^{\infty}\exp(-\frac{2\epsilon t}{\lambda})\,\mathrm{d}t \\ &= \exp(-\frac{\Delta_{j,i}\epsilon}{\lambda})\left(\frac{\Delta_{j,i}\epsilon}{4\lambda}+\frac{1}{2}\right). \end{aligned} \tag{2}$$

2) When $\Delta_{j,i} < 0$, Equation 1 becomes

$$\begin{aligned} &Pr(n_i > n_j + \Delta_{j,i}) \\ &= \frac{\epsilon}{4\lambda}\exp(-\frac{\Delta_{j,i}\epsilon}{\lambda})\int_{-\infty}^{\Delta_{j,i}}\exp(\frac{2\epsilon t}{\lambda})\,\mathrm{d}t \\ &\quad + \frac{\epsilon}{2\lambda}\int_{\Delta_{j,i}}^{0}\exp(\frac{\epsilon t}{\lambda})\,\mathrm{d}t \\ &\quad + \frac{\epsilon}{2\lambda}\int_{0}^{\infty}\exp(-\frac{\epsilon t}{\lambda})\,\mathrm{d}t - \frac{\epsilon}{4\lambda}\exp(\frac{\Delta_{j,i}\epsilon}{\lambda})\int_{\Delta_{j,i}}^{0}\mathrm{d}t \\ &\quad - \frac{\epsilon}{4\lambda}\exp(\frac{\Delta_{j,i}\epsilon}{\lambda})\int_{0}^{\infty}\exp(-\frac{2\epsilon t}{\lambda})\,\mathrm{d}t \\ &= 1 - \frac{1}{2}\exp(\frac{\Delta_{j,i}\epsilon}{\lambda}) + \frac{\Delta_{j,i}\epsilon}{4\lambda}\exp(\frac{\Delta_{j,i}\epsilon}{\lambda}). \end{aligned} \tag{3}$$

Note that both Equation 2 and Equation 3 are decreasing functions of $\Delta_{j,i}$, meaning that it is less likely to pick i over j when $\Delta_{j,i}$ increases.

Next, consider the gradient with largest absolute value and the gradient with the second largest absolute value. Assume the index of these two gradients are m_1 and m_2, respectively. Then by Equation 2 and Equation 3, we have

$$\begin{aligned} P_{m_1} &= \prod_{\substack{j=1,..,N,\\ j\neq m_1}} Pr(|g_{m_1}| + n_{m_1} > |g_j| + n_j) \\ &= \prod_{\substack{j=1,..,N,\\ j\neq m_1}} \left[1 - \frac{1}{2}\exp(\frac{\Delta_{j,m_1}\epsilon}{\lambda}) \right. \\ &\quad \left. + \frac{\Delta_{j,m_1}\epsilon}{4\lambda}\exp(\frac{\Delta_{j,m_1}\epsilon}{\lambda})\right], \end{aligned}$$

and

$$\begin{aligned} P_{m_2} &= \prod_{\substack{j=1,..,N,\\ j\neq m_2}} Pr(|g_{m_2}| + n_{m_2} > |g_j| + n_j) \\ &= \exp(-\frac{\Delta_{m_1,m_2}\epsilon}{\lambda})\left(\frac{\Delta_{m_1,m_2}\epsilon}{4\lambda}+\frac{1}{2}\right) \\ &\quad \prod_{\substack{j=1,..,N,\\ j\neq m_2}} \left[1 - \frac{1}{2}\exp(\frac{\Delta_{j,m_2}\epsilon}{\lambda}) \right. \\ &\quad \left. + \frac{\Delta_{j,m_2}\epsilon}{4\lambda}\exp(\frac{\Delta_{j,m_2}\epsilon}{\lambda})\right]. \end{aligned}$$

Therefore,

$$\begin{aligned} \frac{P_{m_1}}{P_{m_2}} &= \frac{\left[1 - \frac{1}{2}\exp(\frac{\Delta_{m_2,m_1}\epsilon}{\lambda}) + \frac{\Delta_{m_2,m_1}\epsilon}{4\lambda}\exp(\frac{\Delta_{m_2,m_1}\epsilon}{\lambda})\right]}{\exp(-\frac{\Delta_{m_1,m_2}\epsilon}{\lambda})\left(\frac{\Delta_{m_1,m_2}\epsilon}{4\lambda}+\frac{1}{2}\right)} \\ &\quad \underbrace{\prod_{\substack{j=1,..,N,\\ j\neq m_1,m_2}} \frac{\left[1 - \frac{1}{2}\exp(\frac{\Delta_{j,m_1}\epsilon}{\lambda}) + \frac{\Delta_{j,m_1}\epsilon}{4\lambda}\exp(\frac{\Delta_{j,m_1}\epsilon}{\lambda})\right]}{\left[1 - \frac{1}{2}\exp(\frac{\Delta_{j,m_2}\epsilon}{\lambda}) + \frac{\Delta_{j,m_2}\epsilon}{4\lambda}\exp(\frac{\Delta_{j,m_2}\epsilon}{\lambda})\right]}}_{A}. \end{aligned}$$

Note that each component in the product A is larger than 1 since $\Delta_{j,m_1} < \Delta_{j,m_2}$ and equation 3 is a decreasing function. Therefore,

$$\begin{aligned} \frac{P_{m_1}}{P_{m_2}} &\geq \frac{\left[1 - \frac{1}{2}\exp(\frac{\Delta_{m_2,m_1}\epsilon}{\lambda}) + \frac{\Delta_{m_2,m_1}\epsilon}{4\lambda}\exp(\frac{\Delta_{m_2,m_1}\epsilon}{\lambda})\right]}{\exp(-\frac{\Delta_{m_1,m_2}\epsilon}{\lambda})\left(\frac{\Delta_{m_1,m_2}\epsilon}{4\lambda}+\frac{1}{2}\right)} \\ &= \frac{\exp(\frac{\Delta_{m_1,m_2}\epsilon}{\lambda})}{\frac{\Delta_{m_1,m_2}\epsilon}{4\lambda}+\frac{1}{2}} - 1. \end{aligned}$$

■

REFERENCES

[1] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li, "Terngrad: Ternary gradients to reduce communication in distributed deep learning," *arXiv preprint arXiv:1705.07878*, 2017.

[2] N. Agarwal and M. Sharma, "Fraud risk prediction in merchant-bank relationship using regression modeling," *Vikalpa*, vol. 39, no. 3, pp. 67–76, 2014.

[3] R. J. Bolton and D. J. Hand, "Statistical fraud detection: A review," *Statistical science*, pp. 235–249, 2002.

[4] M. F. Zeager, A. Sridhar, N. Fogal, S. Adams, D. E. Brown, and P. A. Beling, "Adversarial learning in credit card fraud detection," in *2017 Systems and Information Engineering Design Symposium (SIEDS)*, April 2017, pp. 112–116.

[5] K. Gai, M. Qiu, and X. Sun, "A survey on fintech," *Journal of Network and Computer Applications*, 2017.

[6] C.-C. Lin, A.-A. Chiu, S. Y. Huang, and D. C. Yen, "Detecting the financial statement fraud: The analysis of the differences between data mining techniques and experts judgments," *Knowledge-Based Systems*, vol. 89, pp. 459–470, 2015.

[7] G. Rushin, C. Stancil, M. Sun, S. Adams, and P. Beling, "Horse race analysis in credit card fraud-deep learning, logistic regression, and gradient boosted tree," in *Systems and Information Engineering Design Symposium (SIEDS), 2017*. IEEE, 2017, pp. 117–121.

[8] S. M. Boker, T. R. Brick, J. N. Pritikin, Y. Wang, T. v. Oertzen, D. Brown, J. Lach, R. Estabrook, M. D. Hunter, H. H. Maes *et al.*, "Maintained individual data distributed likelihood estimation (middle)," *Multivariate behavioral research*, vol. 50, no. 6, pp. 706–720, 2015.

[9] W. Xie, Y. Wang, S. M. Boker, and D. E. Brown, "Privlogit: Efficient privacy-preserving logistic regression by tailoring numerical optimizers," *arXiv preprint arXiv:1611.01170*, 2016.

[10] K. Gai and M. Qiu, "Blend arithmetic operations on tensor-based fully homomorphic encryption over real numbers," *IEEE Transactions on Industrial Informatics*, 2017.

[11] K. Gai, M. Qiu, Z. Xiong, and M. Liu, "Privacy-preserving multi-channel communication in edge-of-things," *Future Generation Computer Systems*, vol. 85, pp. 190–200, 2018.

[12] L. Sweeney, "k-anonymity: A model for protecting privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 05, pp. 557–570, 2002.

[13] A. Narayanan and V. Shmatikov, "Robust de-anonymization of large sparse datasets," in *Security and Privacy, 2008. SP 2008. IEEE Symposium on*. IEEE, 2008, pp. 111–125.

[14] N. Homer, S. Szelinger, M. Redman, D. Duggan, W. Tembe, J. Muehling, J. V. Pearson, D. A. Stephan, S. F. Nelson, and D. W. Craig, "Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays," *PLoS Genet*, vol. 4, no. 8, p. e1000167, 2008.

[15] R. Wang, Y. F. Li, X. Wang, H. Tang, and X. Zhou, "Learning your identity and disease from research papers: information leaks in genome wide association study," in *Proceedings of the 16th ACM conference on Computer and communications security*. ACM, 2009, pp. 534–544.

[16] C. Dwork, "Differential privacy," in *Automata, languages and programming*. Springer, 2006, pp. 1–12.

[17] M. Pathak, S. Rane, and B. Raj, "Multiparty differential privacy via aggregation of locally trained classifiers," in *Advances in Neural Information Processing Systems*, 2010, pp. 1876–1884.

[18] A. Rajkumar and S. Agarwal, "A differentially private stochastic gradient descent algorithm for multiparty classification," in *International Conference on Artificial Intelligence and Statistics*, 2012, pp. 933–941.

[19] J. Hamm, Y. Cao, and M. Belkin, "Learning privately from multiparty data," in *International Conference on Machine Learning*, 2016, pp. 555–563.

[20] S. Song, K. Chaudhuri, and A. D. Sarwate, "Stochastic gradient descent with differentially private updates," in *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*. IEEE, 2013, pp. 245–248.

[21] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of cryptography*. Springer, 2006, pp. 265–284.

[22] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Foundations and Trends in Theoretical Computer Science*, vol. 9, no. 3-4, pp. 211–407, 2014.

[23] P. Kairouz, S. Oh, and P. Viswanath, "The composition theorem for differential privacy," *IEEE Transactions on Information Theory*, 2017.

[24] B. Hitaj, G. Ateniese, and F. Pérez-Cruz, "Deep models under the gan: information leakage from collaborative deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 603–618.

[25] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. ACM, 2015, pp. 1310–1321.

[26] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 308–318.

[27] R. Pascanu, T. Mikolov, and Y. Bengio, "Understanding the exploding gradient problem," *CoRR, abs/1211.5063*, 2012.

[28] T. Mikolov, "Statistical language models based on neural networks," *Presentation at Google, Mountain View, 2nd April*, 2012.

[29] A. Neelakantan, L. Vilnis, Q. V. Le, I. Sutskever, L. Kaiser, K. Kurach, and J. Martens, "Adding gradient noise improves learning for very deep networks," *arXiv preprint arXiv:1511.06807*, 2015.