Privacy-Preserving Collaborative Deep Learning with Irregular Participants

Lingchen Zhao, Yan Zhang, Qian Wang, Member, IEEE, Yanjiao Chen, Member, IEEE, Cong Wang, Member, IEEE and Qin Zou, Member, IEEE

Abstract—With large amounts of data collected from massive sensors, mobile users and institutions becomes widely available, neural network based deep learning is becoming increasingly popular and making great success in many application scenarios, such as image detection, speech recognition and machine translation. While deep learning can provide various benefits, the data for training usually contains highly sensitive information, e.g., personal medical records, and a central location for saving the data may pose a considerable threat to user privacy.

In this paper, we present a practical privacy-preserving collaborative deep learning system that allows users (i.e., participants) to cooperatively build a collective deep learning model with data of all participants, without direct data sharing and central data storage. In our system, each participant trains a local model with their own data and only shares model parameters with the others. To further avoid potential privacy leakage from sharing model parameters, we use functional mechanism to perturb the objective function of the neural network in the training process to achieve $\epsilon$-differential privacy. In particular, for the first time, we consider the possibility that the data of certain participants may be of low quality (called irregular participants), and propose a solution to reduce the impact of these participants while protecting their privacy. We evaluate the performance of our system on two well-known real-world data sets for regression and classification tasks. The results demonstrate that our system is robust to irregular participants, and can achieve high accuracy close to the centralized model while ensuring rigorous privacy protection.

I. INTRODUCTION

In the past few years, deep learning has tremendously revolutionized the fields of machine learning and artificial intelligence, achieving a much better performance than traditional machine learning methods in various applications, e.g., image processing [1], speech recognition [2], [3], cancer analysis [4], [5], and the game of Go [6]. The great success of deep learning is largely owing to the availability of massive data for training the neural networks. In general, the deep learning model will be more accurate if trained with more diverse data, which motivates companies and institutions to collect as much data as possible from their users, usually generated by sensors on users’ personal devices, e.g., GPS, cameras, gyroscopes, and heart rate sensors. From the privacy perspective, however, user-generated data is usually highly sensitive, e.g., location information, personal medical records, and social relationships. To gather these sensitive data at a centralized location will raise serious concerns about privacy leakage. A recent regulation of EU [7] stipulates that companies should carefully collect and use users’ personal data, and users have the right to require the company to permanently “forget” their data. The bill also prohibits any automated individual decision-making (e.g., personal financial situation, personal health condition, and location prediction) based on the data, which may greatly affect the machine learning tasks by companies. In addition, in many sectors especially medical industry, sharing personal data is forbidden by laws or regulations. recently, where multiple parties host some of the data, contribute their computing capacities and finally learn a collective machine learning model which benefits from all parties.

To glean the benefit of machine learning while protecting user privacy, there is a rising interest in designing privacy-assured machine learning algorithms from both academia and industry. Existing solutions regarding traditional machine learning algorithms mainly exploit the intrinsic features of the algorithms, e.g., strictly convex objective functions. Privacy-preserving techniques such as secure multi-party computation or differential privacy have been applied to linear and logistic regression analysis [8], [9], k-means clustering [10], support vector machines [11], and crowd machine learning [12]. In recent years, privacy-preserving deep learning has received much attention from the research community. In [13], cryptographic tool, namely homomorphic encryption, was first applied to convolutional neural networks (CNNs), but the solution is a centralized one and requires extensive computation resources. Subsequently, many works tried to improve the performance of deep learning on encrypted data, such as [14], [15], but all these schemes introduce heavy overheads to the original computation on plaintext data. In [16], only the intermediate representations obtained by a local neural network model are published to hide the private data, but the scheme did not provide a rigorous privacy guarantee. In [17], a differentially private stochastic gradient descent algorithm and a mechanism to accurately track the privacy loss during training were designed, which could train deep neural networks with a modest privacy budget and a manageable model quality, but the scheme still depends on the number of training epochs and some empirical parameters (e.g. lot size, clipping bound). In [18], differential privacy was applied to a specific deep learning model, deep auto-encoder, and sensitivity analysis and noise insertion were conducted on data reconstruction and
cross-entropy error objective functions. In [19], a framework called DSSGD was proposed to ensure differential privacy for distributed deep networks. Some other attacks that try to extract alternative information from the machine learning process are proposed in recent years, e.g., model inversion attacks [20], membership inference attacks [21], and model extraction attacks [22]. In particular, by utilizing generative adversarial network (GAN), it is claimed in [23] that a distributed deep learning approach cannot protect the training sets of honest participants even if the model is trained in a privacy-preserving manner by [19]. However, it is found in [24] that [23] does not truly break the rigorous differential privacy.

In this paper, we investigate the problem of collaborative deep learning with strong privacy protection while maintaining a high data utility. In collaborative deep learning, users (i.e., participants) cooperatively learn a collective deep learning model that can benefit from the data of all users. Our work is most related to [17]–[19], but is quite different in several ways. The proposed schemes in [17] and [18] were not designed for the collaborative deep learning. In [19], participants only share a subset of parameters with the others to reduce communication costs and differential privacy is achieved by inserting noises to truncated weights, but there are some limitations. The consumed privacy budget during the learning process is relatively high for each single parameter. The total privacy budget is proportional to the number of parameters, which may be as high as tens of thousands in deep learning models. The parameter that tunes the fraction of uploaded gradients is used to quantify the privacy, but each pixel in the training data may be revealed by multiple gradients. Furthermore, it is not considered that the data quality of certain participants may be poor, which may degrade the performance of collaborative learning.

To address the above issues, in our proposed mechanism, each participant (e.g., mobile user, medical institution) maintains a local neural network model and a local dataset that may be highly sensitive. Instead of sharing local data with the central server, the participant only uploads the updated parameters of the local model based on the local dataset. The central server derives the global parameters for the collective model according to the updates from all participants. Parameter sharing can prevent direct exposure of the local data, but may indirectly disclose the information of the sensitive data. To solve this problem, we utilize differential privacy [25] to obtain the sanitized parameters to minimize privacy leakage. Unlike [19], where noise is directly injected to the gradients, we apply functional mechanism [26] to perturb the objective function of the neural network, and obtain the sanitized parameters by minimizing the perturbed objective function. In collaborative learning, the quality of data contributed by different participants may be diversified. Different terminal devices or persons may have different capacities to generate the training data, and there may exist unpredictable random errors during data collection and storage. Participants with low quality data are referred to as irregular participants (discussed in detail in Section II-A). To make the learning process fair and non-discriminative, we consider the “data quality” as one of the privacy concerns of participants, which should not be inferred by other participants during the learning process. We adopt exponential mechanism to protect this privacy while effectively learning an accurate model. Our main contributions are summarized as follows.

- We make the first attempt to investigate the problem of privacy-preserving collaborative deep learning taking into account the existence of irregular participants.
- We present a novel scheme called SecProbe, which allows participants to share model parameters and deals with irregular participants by utilizing exponential mechanism. SecProbe can protect the privacy of data quality of each participant while effectively learning an accurate model.
- We derive the approximate polynomial form of the objective function in a neural network with two different loss functions, and use functional mechanism to inject noises to the coefficients to achieve differential privacy without consuming too much privacy budget. We show that it is easy to extend and apply our method to networks with more layers.
- We evaluate the performance of SecProbe on two well-known real-world data sets for regression and classification tasks. The results demonstrate that SecProbe is robust to irregular participants, and achieves high accuracy close to the solution based on the centralized model, while providing a rigorous privacy guarantee.

II. PROBLEM STATEMENT AND PRELIMINARIES

In this section, we introduce our problem statement and some preliminary knowledge of deep learning, differential privacy and functional mechanism used in our design.

A. Problem Statement

In this paper, we consider the problem of privacy-preserving distributed collaborative deep learning. As shown in Figure 1, in our model, each collaborative participant may have their own sensitive data and it would like to learn a model benefiting from both its own data and the others.
In particular, instead of making the assumption that all the participants are “regular”, i.e., the data held by each participant is balanced and has the same or similar quality, we consider a more practical model where there may exist a small group of participants who are “irregular” during some phases of the whole learning process. That is, a portion of data held by irregular participants is not always as accurate as others, and thus their uploaded parameters may disturb the learning accuracy. In our daily lives, irregular participants are common in a collaborative learning system. Consider a typical scenario where several hospitals aim to learn a model together for cancer prediction for patients. There may exist non-negligible gaps in the quality of data among different hospitals since a rich-experienced chief physician with advanced medical devices in a high-rate hospital will be more likely to produce accurate data than a junior physician with low-end of devices in an ordinary hospital. Note that, it does not mean that the data in the ordinary hospital are all and/or always “bad”. Actually, every participant might have some bad data in some phases of their training process when more and more data are being gathered into their local date set, because there are so many possibilities to go wrong in the data generation and storage procedures. Consequently, the existence of irregular participants will bring non-ignorable disturbance during the collaborative training process, which may finally result in an inaccurate or even useless model. In this paper, we aim to reduce the impact of inaccurate data on the accuracy of the learned model in the presence of irregular participants. In our scheme, we assume the irregular participants are not malicious, i.e., they do not deliberately inject false data into the system. Similar to the commonly-used curious-but-honest adversarial model, we refer to irregular participants as reckless-but-honest. In addition, we show that our scheme is secure against active adversaries, i.e., malicious participants, in Section III-D.

We assume the server is honest, and there are two major privacy concerns in our model: 1) privacy of participants’ local data, which may be revealed by the parameters of the local model of each participant and inferred by the server and other participants and 2) privacy of the quality of participants’ data, i.e., it is necessary and important to hide a participant’s real data quality from others in order to preserve its reputation and achieve a non-discriminative collaborative learning environment. The data quality of participants may be inferred by others by observing the global parameters generated by the server.

Keep the above performance requirements and privacy concerns in mind, our ultimate goal is to design a distributed collaborative deep learning system that can jointly learn an accurate model while protecting both the privacy of the participants’ local data and quality privacy of the participants’ data.

B. Deep learning

Broadly speaking, deep learning, based on artificial neural networks, aims to learn and extract high-level abstractions in data and build a network model to describe accurate relations between inputs and outputs. Common deep learning models are usually constructed by multi-layer networks, where non-linear functions are embedded, so that more complicated underlying features and relations can be learned in different layers. Interested readers can refer to thorough surveys or reviews in [27], [28].

There are multiple forms of deep learning models, e.g., multi-layer perceptron (MLP), convolutional neural network (CNN), recurrent neural network (RNN). Different models fit for different types of problems, and among all of those models, MLP is a very common and representative form of deep learning architecture. Specifically, MLP is a kind of feed-forward neural network, where each neuron receives the outputs of neurons from the previous layer. Figure 2 shows a typical MLP with multiple hidden layers. Each neuron has an activation function which is usually non-linear. As shown, for a neuron in a hidden layer, say $j$, the output of the neuron is calculated by the equation $h_j = f(W_j^{(1)}x)$, where $W_j^{(1)}$ is the weight vector which determines the contribution of each input signal to the neuron $j$, $x$ is the input of the model and $f$ is the activation function. The activation function is usually non-linear in order to capture the complicated non-linear relation between the output and input. Typical examples are sigmoid function $f(x) = (1 + e^{-x})^{-1}$, ReLU function $f(x) = \max(0, x)$, and hyperbolic tangent $f(x) = \frac{e^{x} - 1}{e^{x} + 1}$. In our work, we will focus on a MLP model, where ReLU function is applied.

Training a neural network, i.e., learning the parameters (weights) of the network, is a non-convex optimization problem. The typical algorithms used to solve the problem are different types of gradient descent methods [29]. In this paper, we will consider a supervised learning task, e.g., regression analysis, and assume the output of the network is $z$. Suppose the data we use to train the network is a tuple $(x_i, y_i)$, where $x_i$ is used to be the network input and $y_i$ is the label. Consequently, we can use loss (objective) function to measure the difference between the network output and the real training label, e.g., $Error_i = (z_i - y_i)^2$. We then can use back propagation [30] algorithm to propagate the error back to the neurons, compute the contribution of each neuron to this error, and adjust the weights accordingly to reduce the
training error. The adjustment procedure for a weight, say \( w_j \), is \( w_j = w_j - l \frac{\partial \text{Error}}{\partial w_j} \), where \( l \) is the learning rate.

Among various gradient descent algorithms, stochastic gradient descent (SGD) \([21]\) is considered to be especially fit for optimizing highly non-convex problem for its high efficiency and effectiveness. This algorithm brings stochastic factors into the training process, which helps the model to escape from local optimum. For a large training data set, SGD first randomly samples a small subset (mini-batch) of the whole data set, then computes the gradients over the mini-batch and updates the weights, e.g., for weight \( w_j \). After one iteration on the mini-batch, the new weight \( w_j \) is computed by \( w_j = w_j - l \frac{\partial \text{Error}}{\partial w_j} \), where \( \text{Error}_j \) is the loss function computed on the mini-batch \( b \). In our work, we will apply SGD to each participant to train its local model.

C. Differential Privacy

Differential privacy has become a de facto standard privacy model for statistics analysis with provable privacy guarantee, and has been widely used in data publishing \([22, 23]\) and data analysis \([24, 25]\). Intuitively, a mechanism satisfies differential privacy if its outputs are approximately the same even if a single record in the dataset is arbitrarily changed, so that an adversary infers no more information from the outputs about the record owner than from the dataset where the record is absent.

**Definition 1 (Differential Privacy)** \([25]\): A privacy mechanism \( \mathcal{M} \) gives \( \epsilon \)-differential privacy, where \( \epsilon > 0 \), if for any datasets \( D \) and \( D' \) differing on at most one record, and for all sets \( S \subseteq \text{Range}(\mathcal{M}) \),
\[
\Pr[\mathcal{M}(D) \in S] \leq \exp(\epsilon) \cdot \Pr[\mathcal{M}(D') \in S],
\]
where \( \epsilon \) is the privacy budget representing the privacy level the mechanism provides. Generally speaking, a smaller \( \epsilon \) guarantees a stronger privacy level, but also requires a larger perturbation noise.

**Definition 2 (Sensitivity)** \([30]\): For any function \( f : D \rightarrow \mathbb{R}^d \), the sensitivity of \( f \) w.r.t. \( D \) is
\[
\Delta(f) = \max_{D, D' \in \mathcal{D}} || f(D) - f(D') ||_1
\]
for all \( D \) and \( D' \) differing on at most one record.

Laplace mechanism is the most commonly used mechanism that satisfies \( \epsilon \)-differential privacy. Its main idea is to add noise drawn from a Laplace distribution into the datasets to achieve differential privacy, we use FM to firstly perturb \( \phi \) for \( \epsilon \)-differential privacy. Let \( \mathcal{M}_j \) be any two neighboring datasets. Let \( f_D(w) \) and \( f_{D'}(w) \) be the objective functions of any dataset \( D \in \mathcal{D} \),
\[
\mathcal{M}(D, u) = \frac{1}{\epsilon} \sum_{j=0}^{\infty} \sum_{\phi \in \mathcal{F}_j} \lambda_{\Phi_j} \Phi(w).
\]

**Theorem 2 (Exponential Mechanism)** \([27]\): Let \( \Delta u \) be the sensitivity of the utility function \( u : (D \times \mathbb{R}) \rightarrow \mathbb{R} \), the mechanism \( \mathcal{M} \) for any dataset \( D \in \mathcal{D} \),
\[
\mathcal{M}(D, u) = \frac{\exp(-uD, r)}{2\Delta u},
\]
gives \( \epsilon \)-differential privacy.

This theorem implies that Exponential mechanism can make high utility outputs exponentially more likely at a rate that mainly depends on the utility score such that the final output would be approximately optimum with respect to \( u \), and meanwhile give rigorous privacy guarantee.

The composition properties of differential privacy provide privacy guarantee for a sequence of computations.

**Theorem 3 (Sequential Composition)** \([38]\): Let \( \mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_r \) be a set of mechanisms and each \( \mathcal{M}_i \) provides \( \epsilon_i \)-differential privacy. Let \( \mathcal{M} \) be another mechanism that executes \( \mathcal{M}_1(D), \ldots, \mathcal{M}_r(D) \) using independent randomness for each \( \mathcal{M}_i \). Then \( \mathcal{M} \) satisfies \( \sum \epsilon_i \)-differential privacy.

**Theorem 4 (Parallel Composition)** \([38]\): Let \( \mathcal{M}_i \) each provides \( \epsilon_i \)-differential privacy. A sequence of \( \mathcal{M}_i(D) \) over disjoint datasets \( D_i \) provides \( \max(\epsilon_i) \)-differential privacy.

These theorems allow us to distribute the privacy budget among \( r \) mechanisms to realize \( \epsilon \)-differential privacy.

D. Functional Mechanism

Functional mechanism (FM) \([25]\) is a general framework for regression analysis with differential privacy. It can be seen as an extension of the Laplace mechanism which ensures privacy by perturbing the optimization goal of regression analysis instead of injecting noise directly into the regression results.

A typical regression analysis on data set \( D \) returns a model parameter \( \hat{w} \) that minimizes the optimization (objective) function \( f_D(w) = \sum_{i \in D} f(x_i, w) \). However, directly releasing \( \hat{w} \) would raise privacy concern, since the parameters reveal information about data set \( D \) and function \( f_D(w) \). In order to achieve differential privacy, we use FM to firstly perturb the objective function \( f_D(w) \) (by exploiting the polynomial representation of \( f_D(w) \)), and then release the parameter \( \hat{w} \) that minimizes the perturbed objective function \( f_D(w) \).

We assume \( w \) is a vector containing \( d \) values \( w_1, \ldots, w_d \). Let \( \phi(w) \) denote the product of \( w_1, \ldots, w_d \), i.e., \( \phi(w) = w_1^c_1 \cdot w_2^c_2 \cdot \cdots \cdot w_d^c_d \), where \( c_1, \ldots, c_d \in N \). Let \( \Phi_j \) denote the set of all products of \( w_1, \ldots, w_d \) with degree \( j \), i.e., \( \Phi_j = \{w_1^c_1 \cdot w_2^c_2 \cdot \cdots \cdot w_d^c_d | \sum_{i=1}^d c_i = j \} \). By the Stone-Weierstrass Theorem \([39]\), any continuous and differentiable function \( f(w) \) can always be written as a polynomial of \( w_1, \ldots, w_d \), i.e., \( f(x_1, w) = \sum_{|j|=0}^{\infty} \sum_{\Phi \in \Phi_j} \lambda_{\Phi} \Phi(w) \), where \( \lambda_{\Phi} \in \mathbb{R} \) denotes the coefficient of \( \phi(w) \) in the polynomial, and \( J \in [0, \infty] \).

Similarly, we can derive the polynomial function of \( f_D(w) \) as
\[
f_D(w) = \sum_{j=0}^{\infty} \sum_{|j|=0}^{\infty} \sum_{\Phi \in \Phi_j} \lambda_{\Phi} \Phi(w).
\]

**Lemma 1** \([26]\): Let \( D \) and \( D' \) be any two neighboring databases. Let \( f_D(w) \) and \( f_{D'}(w) \) be the objective functions of
regression analysis on $D$ and $D'$, respectively. Then, we have the following inequality

$$\Delta = \sum_{j=1}^{J} \sum_{\phi \in \Phi_j} \left\| \sum_{x_i \in D} \lambda_{\phi x_i} - \sum_{x_i \in D'} \lambda_{\phi x_i'} \right\|_1 \leq 2 \max_{x} \sum_{j=1}^{J} \sum_{\phi \in \Phi_j} \| \lambda_{\phi x} \|_1.$$

In order to achieve $\epsilon$-differential privacy, FM perturbs $f_D(w)$ by injecting Laplace noise into its polynomial coefficients. According to Lemma 2, $f_D(w)$ is perturbed by injecting Laplace noise with scale of $\text{Lap}(\Delta/\epsilon)$ into the polynomial coefficients $\lambda(\phi)$, where $\Delta = 2 \max_{x} \sum_{j=1}^{J} \sum_{\phi \in \Phi_j} \| \lambda_{\phi x} \|_1$. Then we can derive the model parameter $\bar{w}$ which minimizes the perturbed function $\bar{f}_D(w)$. In our work, we propose to utilize functional mechanism in our design to protect the privacy of participants' local data.

### III. SecProbe: Privacy-Preserving Collaborative Deep Learning System

#### A. System Architecture

In Figure 1, we assume there are $N$ participants, and each of them has a sensitive data set for local training. The participants aim to learn a common model, i.e., the architectures of the local models are identical, and the learning objectives are the same. There are many deficiencies and difficulties of collecting all the data from participants in advance and training on the entire data set. Such complicated process of data collection usually incurs high communication overhead, and participants may not be willing to directly upload their data to a third party from the perspective of privacy or business consideration. Therefore, the participants only exchange the parameters (weights) with others, and a server (e.g., a cloud service provider) undertakes the job of communicating with participants, exchanging and storing parameters. In our model, we assume there exists a global model and an auxiliary validation data set on the server. This data set can be very small, and it is easy to be obtained in practice. For example, the data can be collected from participants which have already been expired with no privacy concern or publicly well-tested hand-classified data sets (such as MNIST [40]).

Algorithm 1 gives the high-level steps of SecProbe. The server and participants build their own models and initialize all the parameters before the learning starts. For each communication round, the participants locally train their own models using SGD in a differentially private way. After $I$ times of iteration, the participants upload the perturbed parameters to the server. The server then uses the auxiliary data to compute a utility score for each participant, and then chooses to accept the parameters with certain probability. Next, the averaged model parameters are computed and distributed to each participant for the next round of local training. Note that the participant can terminate its training procedure and drop out from the system at any time if it believes its model is accurate enough, and meanwhile a new participant can also join into the system at anytime. We next describe the detailed procedures of SecProbe on the server side and the participant side, respectively.

#### B. SecProbe: The Server Part

Algorithm 2 gives the pseudocode of SecProbe on the server side. The server first initializes the parameters and waits for the local training results by each participant. When the number of participants who upload their weights to the server reaches a pre-fixed threshold $M$, the server stops receiving the uploaded data and sends a stop signal to notify the other participants that there is no need to upload weights (in step 2). The parameter $M$ is used to control the number of participants that the server plans to utilize per round and meanwhile saves a lot of communication costs. Alternatively, this procedure can also be achieved by randomly assigning a set of $M$ participants at the beginning of each round. These two approaches have their own advantages and both can be used in our design. The former can intrinsically deal with the occurrence of failed uploads, while the latter can save a lot of computation costs on the participant side. Without loss of generality, we adopt the first approach in the description of Algorithm 2.

As discussed above, the existence of irregular participants indicates that the parameters uploaded by them may be disruptive, and it may reduce the accuracy of the global model.
To reduce their effect on the model accuracy, we measure the data quality of these irregular participants by calculating a utility score for each participant. Specifically, the server runs the model on the auxiliary validation data set $D$ with the weights of each of $M$ participants respectively and obtains a utility score for participant $m$. Let $G = [W_1, \ldots, W_m]$ denote the set of uploaded weights, where each item can be used to infer the data quality of each participant. For a regression task, suppose the data set has $d$ samples, we define a scoring function $u(G, D, m)$ as

$$u(G, D, m) = \frac{1}{d} \sum_{i=1}^{d} (1 - \frac{z_i - y_i}{y_i}), \quad (6)$$

where $z_i$ is the output of the model with parameter $G(m)$, and $y_i$ is the real value from the auxiliary data. Without loss of generality, we assume that $y_i$ is in range $[0, 1]$. The scoring function calculates an accuracy score for each participant $m$. However, the sensitivity of Equation (6) is unscalable since the proportion of $|z_i - y_i|$ and $y_i$ is infinite in theory. We observe that the term of the summation in Equation (6) is in range $[-1, 1]$ when $z_i \leq 3y_i$, which normally holds in practice since it is almost impossible for the predicted parameter $z_i$ to deviate from the real value $y_i$, for more than three times.\footnote{We find that the average of $\frac{|z_i - y_i|}{y_i}$, i.e., the mean relative error, is always less than 1 in the experimental results in Section IV.}

Therefore, we add the additional restriction in Equation (6) that the proportion $\frac{z_i}{y_i}$ is no more than 3, and the sensitivity can be bounded by $1$.

Moreover, for a classification task, the scoring function $u$ can be defined as the correct prediction rate directly.

**Lemma 2**: The sensitivity of prediction accuracy for classification is $\Delta u = \frac{1}{2}$.

**Proof 1**: Let $m$ and $n$ denotes the number of correct predictions and the number of samples respectively. The sensitivity $\Delta = m + 1 - m = \frac{n - m}{n(n+1)}$. Since $n \geq 1$ and $n \geq m$, the maximum of $\Delta u$ is $\frac{1}{2}$ when $n = 1$ and $m = 0$.

If the server then chooses the participants only according to the scoring function without any uncertainty, the participants could make inferences easily about which participants hold the low quality data by comparing their own parameters and the new parameters sent from the server. In SecProbe, we utilize exponential mechanism to inject uncertainty into the sampling procedure against this kind of inference. The server samples $K$ participants without replacement such that

$$\text{Pr}[\text{Selecting participant } m] \propto \exp\left(\frac{-u(G, D, m)}{2K\Delta u}\right). \quad (7)$$

**Theorem 5**: The sampling procedure in Algorithm 2 (line 4) satisfies $\epsilon$-differential privacy.

**Proof 2**: Proof sketch. Because sampling one participant consumes $\frac{\epsilon}{2K}$ budget and satisfies $\frac{\epsilon}{2K}$-differential privacy according to Theorem 2 the sampling procedure which samples $K$ participants will satisfy $\frac{K\epsilon}{2K}$-differential privacy. $\square$

Note that Theorem 5 only ensures that this sampling procedure satisfies $\epsilon$-differential privacy at the current training iteration. Due to the composition properties of differential privacy, the privacy level provided by it may degrade during training. We will discuss in detail which privacy level our mechanism will provide during the whole training process in the privacy analysis of this section.

**Remarks.** The above procedure samples a set of participants at an exponential rate based on the scoring function while preventing the sampling procedure from leaking privacy. Therefore, the real quality of uploaded weights from a participant cannot be inferred by others since the new weights are computed on a set of privately-chosen participants, and the system can sample the approximately optimal weights and eliminate the disturbance of irregular participants as much as possible. It is easy to see that the time complexity of the sampling step is $O(KM)$. We can further significantly reduce the running time by implementing the sampling step on a static balanced binary tree as suggested in \footnote{We find that the average of $\frac{|z_i - y_i|}{y_i}$, i.e., the mean relative error, is always less than 1 in the experimental results in Section IV.} \cite{4}. The improved sampling step can run in time $O(M + K \ln(M))$.

After choosing the final accepted weights, the server conducts a model average operation that sets the new global weights to be the average of all the accepted weights. The server finally sends the new weights to every participant, and waits for the next round of uploading. We now briefly explain the reason why model average operation works. The average operation to some extent consistently inherits the procedure of SGD by randomly choosing a mini-batch of the training data to get the sum of errors on the mini-batch and then computing the gradients on the error. The average operation acts as choosing a mini-batch of the data from all the accepted weights and computing the gradients on the overall error. Note that, our experiments show that this operation works well only if the parameters of each participant are randomly initialized by the same seed, which is easy to be implemented, e.g., the participants can download the same initialized parameters from the server to replace their own initialization at the very beginning.

It is worth noting that, while we assume the server is trusted to some extent (i.e., the server can know the data quality.
of each participant after uploading), we can further easily relax this assumption by adopting techniques of anonymous communication with provable security to hide participants from the server by using approaches suggested in [19].

C. SecProbe: The Participant Part

Algorithm [3] presents the pseudocode of SecProbe on the participant side. Each participant has its own local training data set and conducts the standard SGD algorithm to train its local model. Let $W_i$ denote the network weights of participant $i$. To protect the privacy of the participant’s sensitive data being disclosed by $W_i$, the participant applies differential privacy onto the training algorithm to get sanitized weights $W_i$, and uploads it to the server.

To achieve differential privacy, Laplace mechanism is utilized in [19] to directly inject noise to the weights. However, their scheme has to consume too much privacy budget for each weight per epoch in order to achieve acceptable results. Instead of directly injecting noise to the weights $W_i$ in our design we propose to utilize functional mechanism [26] to perturb the objective function of the network, train the model on the perturbed objective function and finally compute the sanitized weights $W_i$. Since the structures of the neural networks may be varied and often depend on specific application scenarios, it is impossible to design a one-size-fits-all differentially private solution for all deep learning models. In this paper, we focus on the most common neural network MLP. Specifically, we first consider a three-layer fully-connected neural network, design algorithms to train the model in a differentially private manner, and then show that more hidden layers can be stacked easily by using our proposed scheme.

The regression problem usually uses mean square error (MSE) as the loss function. Suppose the training set $D$ has a set of $n$ tuples $\kappa_1, \kappa_2, \ldots, \kappa_n$. For each tuple $\kappa_i = (x_i, y_i)$, $X_i$ contains $d$ attributes $(x_{i1}, x_{i2}, \ldots, x_{id})$ and $y_i$ is the label of $\kappa_i$. Without loss of generality, we assume each attribute in $X_i$ and $y_i$ is in the scope of $[0, 1]$, which is easily to be satisfied by data normalization. The MLP takes $X_i$ as input and outputs a prediction $\hat{y}_i$ of $y_i$ as accurate as possible. Then, the objective function can be given by

$$\ell(D, W) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2. \quad (8)$$

Recall the calculations of MLP, we have $z_i = \sigma_1(HW^{(2)})$ and $H = \sigma_2(X^TW^{(1)})$, where $W^{(1)}$ and $W^{(2)}$ are the weight matrixes of the network (as shown in Figure 3). $\sigma_1$ is the sigmoid function, and $\sigma_2$ is the ReLU function. Note that we bound the ReLU function by $[0, 1]$ to avoid introducing an unbounded global sensitivity.

Consequently, for a mini-batch $S$ sampled from the training set $D$, the objective function can be written into the following form

$$\ell(S, W) = \sum_{i=1}^{|S|} (y_i - \hat{y}_i)^2$$

$$= \sum_{i=1}^{|S|} [y_i^2 - 2y_i[1 + e^{-\text{ReLU}(X_i^TW^{(1)})}]] - 1 \quad (9)$$

$$+ [1 + e^{-\text{ReLU}(X_i^TW^{(1)})} - 2].$$

Recall that FM requires the objective function be the polynomial representation of weights $w$, thus we need to approximate Equation $9$ and rewrite it into a polynomial form. Since the first term of Equation $9$ is already in the polynomial form, we only consider the other two terms. To utilize Taylor Expansion to help approximate the functions as suggested in [26], for any $j \in [1, d]$, let $f_{1j}, f_{2j}, g_{1j}$ and $g_{2j}$ be four functions defined as follows

$$f_1 = -2y_i[1 + \exp(-z)]; f_2 = [1 + \exp(-z)]^{-2};$$

$$g_1 = \text{ReLU}(X_i^TW^{(1)})W^{(2)}; g_2 = \text{ReLU}(X_i^TW^{(1)})W^{(2)}. \quad (10)$$

Then, we can rewrite Equation $9$ into the following form

$$\ell(S, W) = \sum_{i=1}^{|S|} [y_i^2 + f_1(g_1(\kappa_i, W)) + f_2(g_2(\kappa_i, W))]. \quad (11)$$

Given the above decomposition of the original loss function, we can then apply Taylor expansion in Equation $11$ and obtain the following equation

$$\ell(S, W) = \sum_{i=1}^{|S|} y_i^2 + 2 \sum_{i=1}^{\infty} \sum_{k=0}^{\infty} \frac{f_i^{(k)}(\gamma_i)}{k!} (g_i(\kappa_i, W) - \gamma_i)^k, \quad (12)$$

where $\gamma_{ij}$ is a real number and without loss of generality we set it to zero for ease of analysis. As can be seen in Equation $12$ the number of polynomial terms is infinite, which may result in an unacceptable large sensitivity. Thus, we propose to truncate Equation $12$ by cutting off all polynomial terms with order larger than 2, i.e., we set $k \in [0, 2]$. Then we can obtain the final polynomial objective function used for training as Equation $13$

Now we are ready to give the following lemma.

**Lemma 3:** Let $S$ and $S'$ be any two neighboring databases. Let $\ell(S, W)$ and $\ell(S', W)$ be the objective functions of MLP.
\[
\hat{\ell}(S, W) = \sum_{i=1}^{|S|} \frac{y_i^2}{2} + \sum_{l=1}^{2} \sum_{k=0}^{2} \frac{f^{(k)}(\gamma_i)}{k!} \left( g_l(\kappa_i, W) - \gamma_i \right)^k
\]

\[
= \sum_{i=1}^{|S|} \left[ (y_i^2 + \sum_{l=1}^{2} f^{(0)}_l(0)) + \sum_{l=1}^{2} \frac{f^{(1)}_l(0)}{2} (\text{ReLU}(X^T W^{(1)}) W^{(2)}) + \frac{2}{2} f^{(2)}_l(0) (\text{ReLU}(X^T W^{(1)}) W^{(2)})^2 \right]
\]

\[
= \sum_{i=1}^{|S|} \left[ y_i^2 - y_i + \frac{1}{4} - \frac{2y_i}{4} (\text{ReLU}(X^T W^{(1)}) W^{(2)}) + \frac{1}{16} (\text{ReLU}(X^T W^{(1)}) W^{(2)})^2 \right] \tag{13}
\]

on \(S\) and \(S'\) respectively. Then, the global sensitivity of the objective function \(\hat{\ell}\) over \(S\) and \(S'\) is

\[
\Delta \leq \frac{1}{2} b + \frac{1}{8} b^2,
\]

where \(b\) is the number of hidden units in the hidden layer.

**Proof:** Without loss of generality, we assume that \(S\) and \(S'\) differ in the last tuple, \(\kappa_{i|S|}(\kappa'_{i|S|})\). According to Lemma 1, we have

\[
\Delta \leq 2 \max_k \left( \frac{1}{2} \sum_{p=1}^{b} h_p + \frac{1}{16} \sum_{p=1, q=1}^{b} h_p h_q \right)
\]

\[
\leq 2 \left( \frac{1}{2} b + \frac{1}{16} b^2 \right) = \frac{1}{2} b + \frac{1}{8} b^2,
\]

where \(h\) is the value of hidden neurons at the hidden layer.

As can be seen, the sensitivity of the objective function \(\hat{\ell}(S, W)\) only depends on the model structure, which is independent with the cardinality of the data set \(S\). Finally, we inject Laplace noise with scale \(\frac{\Delta}{2}\) to the coefficients of \(\hat{\ell}(S, W)\) and obtain the perturbed objective function \(\hat{\ell}(S, W)\), which satisfies \(e\)-differential privacy.

The classification problem usually adopts cross-entropy error as the loss function. We take a compact CNN to solve the binomial classification problem with one convolution layer, one pooling layer and one fully-connected layer as an example. We choose sigmoid as the activation function. Similar to the regression problem, the objective function is

\[
\ell(\kappa_i, W) = -y_i \log z_i + (1 - y_i) \log(1 - z_i). \tag{14}
\]

Equation (14) can also be decomposed to four functions as follows

\[
f_1 = y_i \log[1 + \exp(-z_i)];
\]

\[
f_2 = (1 - y_i) \log[1 - (1 + \exp(-z_i))] - 1; \tag{15}
\]

\[
g_1 = g_2 = \text{Conv}(\kappa_i) W^{(2)},
\]

note that \(\text{Conv}(\kappa_i)\) represents the output of the previous convolution layer, and \(W^{(2)}\) is the weight matrix of the fully-connected layer.

The loss function can be rewritten as Equation (16)

\[
\ell(S, W) = - \sum_{i=1}^{|S|} [f_1 (g_1(\kappa_i, W)) + f_2 (g_2(\kappa_i, W))]. \tag{16}
\]

The expansion form of Equation (16) is

\[
\tilde{\ell}(S, W) = - \sum_{i=1}^{2} \sum_{k=0}^{2} \sum_{l=1}^{2} \frac{f^{(k)}(\gamma_i)}{k!} \left( g_l(\kappa_i, W) - \gamma_i \right)^k. \tag{17}
\]

Moreover, the functional mechanism can also be applied to other types of loss functions (e.g., huber-loss function), other activation functions (e.g., hyperbolic tangent), and other types of networks (e.g., Auto-Encoder or RNN) with certain adaptations, which are beyond the scope of this paper.
\[ \hat{\ell}(S, W) = - \sum_{i=1}^{[S]} \sum_{l=1}^{2} \sum_{k=0}^{2} \left[ \frac{f_l^{(k)}(\gamma_l)}{k!} (g_l(\kappa, W) - \gamma_l)^k \right] = \sum_{i=1}^{[S]} \left[ \log 2 + \left( \frac{1}{2} - y_i \right) \text{Conv}^{-1}(\kappa_i) W^{(2)} + \frac{1}{8} \text{Conv}^{-1}(\kappa_i) W^{(2)^2} \right] \] (18)

D. Security Analysis

Let the privacy budget used in sampling participants and perturbing objective functions be \( \epsilon_1 \) and \( \epsilon_2 \) respectively. Since the training procedure on each participant strictly follows the Functional Mechanism, the parameters computed from the perturbed objective function satisfy \( \epsilon \)-differential privacy in each training iteration. Let \( S_i \) be the training batch of an iteration. Since every batch is disjoint from each other in a training epoch (e.g. \( S_i \) and \( S_{i-1} \) contains different tuples sampling randomly from the training data), where an epoch is one full training process that consists of several iterations covering the whole training data, we can conclude that the training process on each participant ensures \( \epsilon_2 \)-differential privacy in each epoch according to Theorem 4.

Recall that the sampling procedure in Algorithm 2 also ensures \( \epsilon_1 \)-differential privacy on each sampling step. Since it can be seen that each step of sampling protect the privacy of part of the training data’s quality, we can conclude that the sampling procedure in Algorithm 2 satisfies \( \epsilon_1 \)-differential privacy in each epoch.

Note that the two procedures above address two different privacy concerns respectively. In case of passive adversaries, the procedure of each participant aims to protect the privacy of the training data, which focuses on each single record of the training data, while the procedure on the server aims to protect the privacy of the data quality, which takes all the corresponding records as a whole. Therefore, we can finally conclude that our scheme SecProbe satisfies \( \max(\epsilon_1, \epsilon_2) \)-differential privacy in each training epoch.

We further consider stronger adversaries. If the adversaries are active, they may perform two kinds of behaviors: 1) send fake parameters to the server; 2) steal the parameters from the communication process directly. For the first malicious behavior, thanks to the exponential mechanism that we have introduced, it is almost impossible for the fake parameters to significantly affect the model, thus there is no negative effect on the training process, and the adversaries cannot infer the data quality of other participants. For the second malicious behavior, the adversary may eavesdrop on channels between honest participants and the server. Some effective cryptography tools can be used to encrypt the parameters (e.g., AES) and verify the received data (e.g., SHA-256) to ensure communication security. Therefore, our scheme is robust and secure when facing active adversaries, i.e., malicious participants.

IV. Experimental Evaluation

In this section, we evaluate the performance of SecProbe on a real-world data set for regression analysis. All experiments are conducted on a machine with Intel Core i5-4460S CPU 2.9GHz and 12GB RAM, running Ubuntu 14.04.

| Communication round | 443 | 190 | 152 | 193 |
|---------------------|-----|-----|-----|-----|
| \( I \)             | 20  | 50  | 100 | 1000 |

TABLE I: The effect of iterations \( I \) which controls the frequency of updates. Each entry in the table gives the necessary number of communication rounds to achieve MRE of 0.15 (\( N = 60, M = 30, K = 30, P = 0, \) and \( \epsilon = 1 \)).

A. Datasets

For the regression task, we use the data set from Integrated Public Use Microdata Series [32], named US, which contains 600,000 census records collected in US. There are 15 attributes in the data set, namely, Sex, Age, Race, Education, Field of Degree, Marital Status, Family Size, Number of Children, Hours Work per Week, Ownership of Dwelling, Number of Children, Number of Rooms, Private Health Insurance, Living Difficulty, Annual Income. Among all these attributes, there are 6 attributes that are categorical, including Race, Education, Field of Degree, Marital Status, Private Health Insurance, Living Difficulty. For an attribute that can only be two possible values (e.g., male and female for sex), we set it to be 0 or 1. For the remaining, we follow the common practice in machine learning to transform these attributes by one-hot encoding. We then normalize the other numeric attributes into the scope of \([0, 1]\). Specifically, for the Annual Income, we apply log transformation before normalization in order to obtain a relatively stable distribution. After these transformations, our data set now has 20 dimensions.

We randomly sample 90,000 records to be the test data set, and 10,000 records to be the auxiliary validation data set on the server. The remaining 500,000 records are randomly divided into \( N \) parts, where \( N \) is the number of participants. The data is already shuffled before training.

We focus on a regression task predicting the value of Annual Income by using the other attributes as the input. The accuracy of the model is measured by mean relative error (MRE),

\[ \text{MRE} = \frac{1}{n} \sum_{i=1}^{n} \frac{|z_i - y_i|}{y_i}, \] (19)

where \( y_i \) is the real value, \( z_i \) is the predicted value produced by the network, and \( n \) is the number of tuples in the test data set.

For the binomial classification task, we use the MNIST data set as the benchmark, which consists of 28x28 images of handwritten digits with 60,000 training samples and 10,000 test samples. We use prediction accuracy to evaluate the performance of the model.

B. Experimental Setup

We use the popular neural network architectures: multi-layer perceptron (MLP) with three fully-connected layers. For the
For the classification task, we use a compact CNN with only one convolution and one hidden layer, the activation functions of the output layer is sigmoid function, and the number of neurons in the hidden layer is 128. The weights of the models are randomly initialized by normal distribution (with mean 0 and standard deviation 1).

For the classification task, we use a compact CNN with only one convolution and one hidden layer, the activation functions of the output layer is sigmoid function, and the number of neurons in the fully-connected layer is 128. The other hyperparameters are the same as the regression task.

Since the approaches proposed in [18] and [17] are not specially designed for collaborative learning, we mainly compare all results with DSSGD [19], an existing work on privacy-preserving collaborative deep learning, and two baseline approaches. The first is the centralized training on the entire data set, which is the basic approach that does not consider privacy concerns and ought to have the best performance of the model accuracy. The second is stand-alone training, which trains solely on local data set without collaboration. We call these two baselines Centralized and Stand-alone, respectively. All approaches are implemented on TensorFlow [43], a popular deep-learning library developed by Google. For SecProbe, we set the privacy budgets used in participants’ sampling and perturbing objective functions to be the same. We fine-tune the parameters in DSSGD according to [19] and use the settings with the best performance (the parameter download ratio \( \theta_d = 1 \), gradient bound \( \gamma = 0.001 \), gradient selecting threshold \( \tau = 0.0001 \)).

To simulate the irregular participants, we randomly choose half of the participants and replace \( P \) fraction of their data with random noise in the scope of \([0,1]\). We vary \( P \) to evaluate the robustness of SecProbe against irregular participants.

C. Results

The effect of \( I \). Table I shows the effect of parameter \( I \). The results show that a larger \( I \) will speed up the training convergence by increasing the computation loads on each participant. However, a too large \( I \) will slow down the training convergence since the collaboration decreases. Based on these results, we set \( I \) to be 100 in the following experiments.

Training convergence. Figure [4] and Figure [5] shows the training convergence of all schemes for regression and classification tasks respectively. We vary the number of participants \( N \) in SecProbe and DSSGD, and set \( M = N, K = M, \epsilon = 1, P = 0 \). The y-axis is the performance of trained model, and the x-axis denotes the number of communication rounds. The results show that although Centralized achieves the best accuracy in the end, SecProbe can achieve almost the same accuracy with a higher convergence rate, while providing a rigorous privacy guarantee. Note that our scheme also has a better performance than DSSGD both on convergence rate and model accuracy in the regression task, and almost the
same performance in the classification task with more rigorous privacy guarantee. The main reason is that only perturbing the uploaded gradients is not enough to preserve the training data, and DSSGD consumes too much privacy budget in perturbing all the gradient values. Moreover, the noise directly injected on each gradient independently may also make the training process unstable. In addition, SecProbe can always reach a better final accuracy than DSSGD, and achieve a comparable convergence rate with DSSGD when the uploaded ration is 1 in CNN.

The effect of parallelism. The parameter $M$ controls the number of participants the server choose to utilize per round, which can also be regarded as parallelism degree. We vary $M$ to be $(0.1N, 0.3N, 0.5N, 0.7N, 1.0N)$ and set $K = M$, $\epsilon = 1$, and $P = 0$. Note that the total size of training data set does not change with different values of $M$. As can be seen in Figure 6, the increase of parallelism will speed up the convergence of training, and it leads to a more accurate model while increasing the communication loads. On the contrary, the decrease of parallelism will reduce the number of accesses for each data point. In brief, the size of $M$ affects the amount of data for training in each communication round and the convergence speed. It is good to see that, when $M = 0.1N$, it can still achieve relatively accurate results. Based on the above results, we choose $M = 0.5N$ for the following experiments to strike a good balance between efficiency and convergence rate.

The effect of auxiliary validation set. To show the effect of the auxiliary validation set on the system performance, we re-design the training set and the test set to simulate one or multiple special participants whose hypotheses are not include at the server. More specifically, we exclude all samples whose one attribute is within a certain range from the original datasets, and put these excluded data to the special participants and a special test set. Here we choose the attribute Age due to its numeric form and its impact on the income. We remove all samples with $Age \leq 0.25$ or $Age \geq 0.7$ from the original datasets. We set the number of special participants as 1, 0.1N, and 0.2N respectively, and demonstrate the results in Figure 7. We can see that the performance of the special test set is significantly worse than that of the regular test set. With more special participants, the performance of the special test set will improve due to the increasing contribution of the special participants to the collaborative learning process. In fact, if a participant’s data is completely random noise, it is almost impossible to be chosen by the server to calculate the average thanks to the exponential mechanism. A special participant with high-quality dataset that contains new findings that might be misjudged can also play a role in improving the model.

The robustness against irregular participants. Figure 9 shows the results of the robustness against irregular participants. We choose half of the participants and replace $P$ proportion of their data with random noise, and set $M = 0.5N$, $K = 0.5M$, and $\epsilon = 1$. We vary the total number of participants $N$ and the proportion $P$ of the noise data, which means that one half of the participants are irregular with $P$ fraction of their data to be random noise. We set $N = (30, 60, 100)$ and
Fig. 8: Accuracy vs. privacy budget $\epsilon$ ($N = 60$, $M = 30$, $K = 15$, and $P = 0.2$).

$P = (0.2, 0.4, 0.6)$. Correspondingly, for Centralized, we set $P = (0.1, 0.2, 0.3)$. For the number of sampling participants $K$, we set it to be the half of $M$, which follows the assumption that the majority of the participants are regular. As can be seen in Figure 8, the model accuracy of DSSGD, Centralized and Stand-alone degrade quickly with the increase of noise, since they all lack in designing mechanisms against irregular participants. Meanwhile, Our approach SecProbe achieves very high accuracy which is almost the same as the performance of the case with no irregular participants, and it is also robust against the proportion of noise. The experimental results validate the effectiveness of our scheme.

Accuracy vs. privacy. We evaluate the effect of different values of privacy budget $\epsilon$ on the accuracy of the neural network. Figure 9 shows the results compared with the competitors. The $x$-axis represents the privacy budget per training epoch (an epoch contains several iterations over all training samples). It is shown that, a larger value of $\epsilon$ results in high accuracy while giving lower privacy guarantee. SecProbe can achieve almost the same results of Centralized and outperform Stand-alone when $\epsilon \geq 0.1$.

V. CONCLUSIONS

In this paper, we took the first step to investigate the problem of privacy-preserving collaborative deep learning system while considering the existence of irregular participants, and presented a new scheme called SecProbe. SecProbe utilizes exponential mechanism and functional mechanism to protect both the privacy of the participants’ data and the quality of their data, the two major privacy concerns in such a system. The experimental results demonstrate that our system is robust to irregular participants, and can achieve high accurate results which is close to the centralized model, while providing rigorous privacy guarantee.

REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” arXiv preprint arXiv:1512.03385, 2015.

[2] A. Graves, A.-r. Mohamed, and G. Hinton, “Speech recognition with deep recurrent neural networks,” in Proc. of ICASSP’13. IEEE, 2013, pp. 6645-6649.

[3] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath et al., “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups,” IEEE Signal Processing Magazine, vol. 29, no. 6, pp. 82–97, 2012.

[4] R. Fakoor, F. Ladhak, A. Nazi, and M. Huber, “Using deep learning to enhance cancer diagnosis and classification,” in Proc. of ICML’13, 2013.

[5] M. Liang, Z. Li, T. Chen, and J. Zeng, “Integrative data analysis of multi-platform cancer data with a multimodal deep learning approach,” IEEE/ACM Transactions on Computational Biology and Bioinformatics, vol. 12, no. 4, pp. 928–937, 2015.

[6] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot et al., “Mastering the game of go with deep neural networks and tree search,” Nature, vol. 529, no. 7587, pp. 484-489, 2016.

[7] “General data protection regulation,” https://en.wikipedia.org/wiki/General_Data_Protection_Regulation, 2016.

[8] “General data protection regulation,” https://en.wikipedia.org/wiki/General_Data_Protection_Regulation, 2016.

[9] K. Chaudhuri and C. Monteleoni, “Privacy-preserving logistic regression,” in Proc. of NIPS’09, 2009, pp. 289–296.

[10] G. Jagannathan and R. N. Wright, “Privacy-preserving distributed k-means clustering over arbitrarily partitioned data,” in Proc. of KDD’05. ACM, 2005, pp. 593–599.

[11] B. I. Rubinstein, P. L. Bartlett, L. Huang, and N. Taft, “Learning in a large function space: Privacy-preserving mechanisms for svm learning,” arXiv preprint arXiv:0911.5708, 2009.

[12] J. Hamm, A. C. Champion, G. Chen, M. Belkin, and D. Xuan, “Crowdml: A privacy-preserving learning framework for a crowd of smart devices,” in Proc. of ICDCS’15. IEEE, 2015, pp. 11–20.

[13] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, “Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy,” in Proc. of ICML’16, 2016, pp. 201–210.

[14] J. Liu, M. Jiutti, Y. Lu, and N. Asokan, “Oblivious neural network predictions via minnon transformations,” in Proc. of CCS’17. ACM, 2017, pp. 619–631.

[15] P. Mohassel and Y. Zhang, “Secureml: A system for scalable privacy-preserving machine learning,” in Proc. of S&P’17. IEEE, 2017, pp. 19–38.

[16] M. Li, L. Lai, N. Suda, V. Chandra, and D. Z. Pan, “PrivyNet: A flexible framework for privacy-preserving deep neural network training,” arXiv preprint arXiv:1709.06161, 2017.

[17] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, “Deep learning with differential privacy,” in Proc. of CCS’16. ACM, 2016, pp. 308–318.
18. N. Phan, Y. Wang, X. Wu, and D. Dou, “Differential privacy preservation for deep auto-encoders: an application of human behavior prediction,” in *Proc. of AAAI’16*, vol. 16, 2016, pp. 1309–1316.
19. R. Shokri and V. Shmatikov, “Privacy-preserving deep learning,” in *Proc. of CCS’15*. ACM, 2015, pp. 1310–1321.
20. M. Fredrikson, S. Jha, and T. Ristenpart, “Model inversion attacks that exploit confidence information and basic countermeasures,” in *Proc. of CCS’15*. ACM, 2015, pp. 1322–1333.
21. R. Shokri, M. Stronati, C. Song, and V. Shmatikov, “Membership inference attacks against machine learning models,” in *Proc. of S&P’17*. IEEE, 2017, pp. 3–18.
22. F. Tramer, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, “Stealing machine learning models via prediction apis.” in *Proc. of USENIX Security’16*, 2016, pp. 601–618.
23. B. Hitaj, G. Ateniese, and F. Pérez-Cruz. “Deep models under the gan: information leakage from collaborative deep learning,” in *Proc. of CCS’17*. ACM, 2017, pp. 603–618.
24. “Deep learning and differential privacy,” https://github.com/frankmcsherry/blog/blob/master/posts/2017-10-27.md, 2016.
25. C. Dwork, “Differential privacy,” in *Proc. of ICALP’06*, 2006, pp. 1–12.
26. J. Zhang, Z. Zhang, X. Xiao, Y. Yang, and M. Winslett, “Functional mechanism: regression analysis under differential privacy,” *Proc. of VLDB* 12, vol. 5, no. 11, pp. 1364–1375, 2012.
27. J. Schmidhuber, “Deep learning in neural networks: An overview,” *Neural Networks*, vol. 61, pp. 85–117, 2015.
28. L. Deng, “A tutorial survey of architectures, algorithms, and applications for deep learning,” *APSIPA Transactions on Signal and Information Processing*, vol. 3, p. e2, 2014.
29. M. Avriel, *Nonlinear programming: analysis and methods*. Courier Corporation, 2003.
30. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.
31. T. Zhang, “Solving large scale linear prediction problems using stochastic gradient descent algorithms,” in *Proc. of ICML’04*. ACM, 2004, p. 116.
32. Q. Wang, Y. Zhang, X. Lu, Z. Wang, Z. Qin, and K. Ren, “Real-time and spatio-temporal crowd-sourced social network data publishing with differential privacy,” *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 4, pp. 591–606, 2018.
33. R. Chen, Q. Xiao, Y. Zhang, and J. Xu, “Differentially private high-dimensional data publication via sampling-based inference,” in *Proc. of KDD’15*. ACM, 2015, pp. 129–138.
34. B. Yang, I. Sato, and H. Nakagawa, “Bayesian differential privacy on correlated data,” in *Proc. of SIGMOD’15*. ACM, 2015, pp. 747–762.
35. D. Proserpio, S. Goldberg, and F. McSherry, “Calibrating data to sensitivity in private data analysis: a platform for differentially-private analysis of weighted datasets,” *Proc. of VLDB* 14, vol. 7, no. 8, pp. 637–648, 2014.
36. C. Dwork, F. McSherry, K. Nissim, and A. Smith, “Calibrating noise to sensitivity in private data analysis,” in *Theory of cryptography*, 2006, pp. 265–284.
37. F. McSherry and K. Talwar, “Mechanism design via differential privacy,” in *Proc. of FOCS’07*. IEEE, 2007, pp. 94–103.
38. F. D. McSherry, “Privacy integrated queries: an extensible platform for privacy-preserving data analysis,” in *Proc. of SIGMOD’09*. ACM, 2009, pp. 19–30.
39. W. Rudin, *Principles of mathematical analysis*. McGraw-Hill New York, 1964, vol. 3.
40. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of The IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
41. R. Bhaskar, S. Laxman, A. Smith, and A. Thakurta, “Discovering frequent patterns in sensitive data,” in *Proc. of KDD’10*. ACM, 2010, pp. 503–512.
42. “Minnesota population center. public use microdata series, international: Version 6.4. university of minnesota, 2015.” https://international.ipums.org
43. “Tensorflow,” https://www.tensorflow.org