2019 IEEE International Conference on Big Data (Big Data)

Profit Allocation for Federated Learning

Tianshu Song, Yongxin Tong*, Shuyue Wei
SKLSDE Lab, BDBC and IRI, Beihang University
{songts, yxtong, weishuyue}@buaa.edu.cn

***Abstract*—Due to stricter data management regulations such as General Data Protection Regulation (GDPR), traditional production mode of machine learning services is shifting to federated learning, a paradigm that allows multiple data providers to train a joint model collaboratively with their data kept locally. A key enabler for practical adoption of federated learning is how to allocate the profit earned by the joint model to each data provider. For fair profit allocation, a metric to quantify the contribution of each data provider to the joint model is essential. Shapley value is a classical concept in cooperative game theory which assigns a unique distribution (among the players) of a total surplus generated by the coalition of all players and has been used for data valuation in machine learning services. However, prior Shapley value based data valuation schemes either do not apply to federated learning or involve extra model training which leads to high cost. In this paper, given n data providers with data sets $D_1, D_2, \cdots, D_n$, a federated learning algorithm $\mathcal{A}$ and a standard test set T, we propose the contribution index, a new Shapley value based metric fit for assessing the contribution of each data provider for the joint model trained by federated learning. The contribution index shares the same properties as Shapley value. However, direct calculation of the contribution index is time consuming, since a large number of joint models with different combinations of data sets are required to be trained and evaluated. To solve this problem, we propose two gradient based methods. The idea is to reconstruct approximately the models on different combinations of the data sets through the intermediate results of the training process of federated learning so as to avoid extra training. The first method reconstructs models by updating the initial global model in federated learning with the gradients in different rounds. Then it calculates the contribution index by the performance of these reconstructed models. The second method calculates contribution index in each round by updating the global model in the previous round with the gradients in the current round. Contribution indexes of multiple rounds are then added with elaborated weights to get the final result. We conduct extensive experiments on the MNIST data set in different settings. The results demonstrate that the proposed methods can approximate the exact contribution index effectively and achieve a time speed up of up to 2x-100x compared with the exact calculation and other baselines extended from existing work.**

***Index Terms*—Federated Learning, Incentive Mechanism, Shapley Value**

I. Introduction

In recent years, machine learning is more and more popular and applied widely in both academic and industrial community. In traditional production mode of machine learning services, data from different sources is first collected and then trained to achieve effective models. However, more and more strict regulations on the use of data such as the General Data Protection Regulation (GDPR) [1], which is a regulation in EU law on data protection and privacy for all individual citizens of the European Union and the European Economic Area, bring challenges on the above production mode and affect the application of machine learning techniques. Specifically, data used in machine learning usually comes from large-scale enterprises and generated by the users. The data often has sensitive information about the users, such as their identification information, characters, habits, credits and even health information. To comply with data regulations such as GDPR, it is becoming more and more infeasible to collect data from different entities together to training a machine learning model.

To solve the above challenge, federated learning is proposed first by Google [2], which enables mobile phone users to train a global model under the interaction with the server. Users can keep their own data locally and only upload a sub-models trained by their own data. Yang *et al.* enrich the concept of federated learning [3]. They categorize federated learning into horizontal federated learning and vertical federated learning. Horizontal federated learning means that the data from different providers has the same attributes. Vertical federated learning means that the data from different providers has different attributions. Besides, Yang *et al.* also propose a practical scenario of federated learning, which we call enterprise federated learning. Different from the scenario proposed by Google in [2], where the number of data providers is big and the data size provided by each data provider is small, in enterprise federated learning, the data providers are entities such as companies and institutions who have big size of data. Meanwhile, the number of data providers in enterprise federated learning is usually small.

To make federated learning practical in industrial scenario, it is indispensable to evaluate the contribution of different data providers and thus the profit earned by the federated model can be allocated. Based on evaluating the contribution reasonably, incentive mechanisms can be designed to attract more data owners to join in federated learning.

Shapley value [4] is a classical concept in game theory [5], which is proposed by and named in honour of Lloyd Shapley. Shapley value is a solution in cooperative game theory [5] which assigns a unique distribution of a total profit generated by the coalition of all players with a collection of desirable properties. Recently, some research use Shapley value to evaluate the contribution of different data points in a data set when training a machine learning model. For example, in [6],

978-1-7281-0858-2/19$31.00 © 2019 IEEE

Ghorbani *et al.* propose the concept of data Shapley value, which has the same form of Shapley value. They propose two heuristic methods to calculate the data Shapley value of each data points efficiently. Jia *et al.* propose theories and methods to optimize the efficiency of the calculation of Shapley value of data points for specific scenarios [7] or machine learning algorithms [8]. Some of the above methods can be extended to work in the scenario of federated learning, where data should be kept locally. However, the above methods involve extra rounds of model training on combinations of data sets from different data providers. The cost of extra rounds of model training is very heavy in the scenario of enterprise federated learning because the data volume is usually very large. Thus, to measure the contribution of different data providers in enterprise federated learning effectively and efficiently, new methods should be designed.

In this paper, we focus on the scenario of horizontal enterprise federated learning. Based on the concept of Shapley value, we define formally the Contribution Index (CI) of different data providers in a federated learning task. We propose two efficient methods to calculate the CIs. The key idea of our methods is that we only need to records intermediate results during the training process of federated learning and use these intermediate results to calculate the CIs approximately. The first method reconstructs models by updating the initial global model in federated learning with the gradients in different rounds and calculates the contribution index by the performance of these reconstructed models. The second method calculates contribution index in each round by updating the global model in the previous round with the gradients in the current round and then aggregate the contribution indexes of multiple rounds to get the final result. Since no extra training process is needed, our methods are very efficient.

To summarize, we have the following contribution.

- Based on Shapley value, we formally define the concept of Contribution Index (CI) to quantify the contribution of data providers in a horizontal federating learning task.
- We design two efficient algorithms to calculation the CIs approximately. Our methods only leverage the intermediate results of the federating learning procedure and involve no extra model training.
- We conduct extensive experiments on different setting to verify the effectiveness and efficiency of the proposed methods on the MNIST data set. The results show that our methods can approximate the exact CI very well and achieve a time speed up of up to 2x-100x.

The remaining of the paper is organized as follows. We review related works in Sec. II, introduce preliminary and definition in Sec. III. Our methods are introduced in Sec. IV. We demonstrate the experimental results in Sec. V and conclude the paper in Sec. VI.

II. Related Work

We review related work from two categories: federated learning and Shapley value based data evaluation.

A. Federated Learning

In recent years, data privacy has been receiving more and more attention. After the new data regulations such as General Data Protection Regulation (GDPR) [1], the traditional production mode of large scale of machine learning services where data from different entities is collected first and then trained carries a high level of risk.

To deal with this new challenge, federated learning is first proposed by Google [2]. In [2], the data providers train models locally on their own data to protect their data privacy and submit gradients to the server to build a global model jointly. Following [2], researchers propose other optional privacy preserving techniques such as Secure Multi-party Computation [9]–[11], Homomorphic Encryption [3], [12], [13] and Differential Privacy [14]–[17].

A typical application scenario of Google's framework [2] is training a model trough federated learning on the data of millions of Android mobile phones [18], [19]. We call this scenario as individual federated learning. The characteristic of individual federated learning is that the number of data providers is very large, but the size of number provided by each data provider is small. In addition to Google [2], researchers have also proposed some variant frameworks of federated learning, such as Nonparametric Federated Learning [20] and Agnostic Federated Learning [21]. In this paper, we adopt the training framework proposed by Google [2], [22].

Yang *et al.* enrich the concept of federated learning [3]. They categorize federated learning into horizontal federated learning and vertical federated learning. In horizontal federated learning, the data of different entities has the same attributions. In this sense, Google's pioneering work [2] can be viewed as horizontal federated learning. In vertical federated learning, the data of different entities has different attributions. Besides, Yang *et al.* [3] also propose a new and practical scenario of federated learning, where the entities may be companies of institutions. For example, some hospitals would like to train a medical imaging classification model jointly. We call this scenario as enterprise federated learning. The characteristic of enterprise federated learning is that the number of data providers is usually small, but the size of number provided by each data provider is very large.

To summarize, according to whether the attributes of different data sets are same, federated learning can be categorized into vertical federated learning and horizontal federated learning. According to the application scenario, federated learning can be categorized into individual federated learning and enterprise federated learning. In this paper, we focus on horizontal enterprise federated learning.

TensorFlow is an end-to-end open source platform for machine learning [23]. TensorFlow Federated [24] is an open-source framework for federated learning, whose training method is based on [2].

In this paper, we define a concept called Contribution Index (CI) which is used to measure the contributions of different data providers in horizontal federated learning. Based on the CIs of the data providers, the profit earned by the joint model

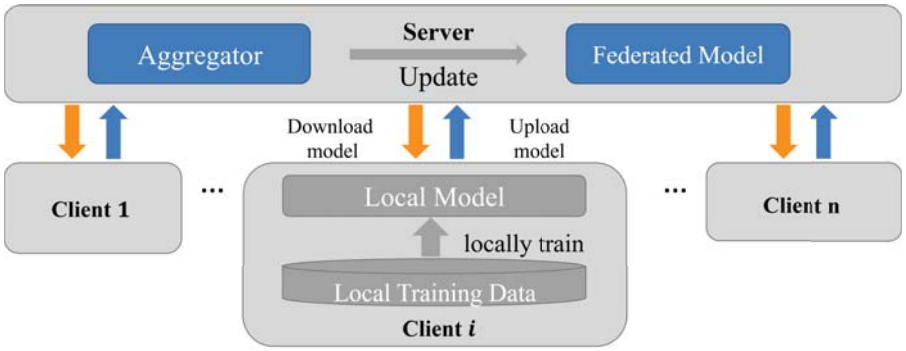

Fig. 1: Workflow of Federated Learning

can be allocated. We follow the training framework proposed in [2], [22] and implement our methods using TensorFlow Federated [24].

B. Shapley Value based Data Evaluation

Shapley value, named in honour of Lloyd Shapley, was first introduced in 1953 [4] and is a solution concept in cooperative game theory [5]. It assigns a unique distribution of a total profit generated by the coalition of all players with a collection of desirable properties.

Ghorbani *et al.* [6] define a concept called data Shapley value, which is based on Shapley value [4] and is used to quantify the the contribution of individual data points to a learning task. To improve the efficiency of the calculation of data Shapley value, they propose two heuristic methods. The first one is called Truncated Monte Carlo Shapley (TMC-Shapley) which samples different orders of data points trained in a machine learning task to approximate the exact data Shapley value. This method can be extended in our scenario and is compared in our experiments. The second one is called Gradient Shapley (G-Shapley), which cannot be integrated in the framework of the federated learning framework adopted in this paper.

Jia *et al.* [7] also use Shapley value to measure the valuation of data points in a data set and focus on improving the efficiency. They develop a repertoire of techniques for estimating the Shapley value of data points in different scenarios. One of their methods, called Group Testing Based SV Estimation, is another sample-based method to calculate Shapley value efficient. We extend this method to our problem and compare with it in our experiments. Based on the empirical observation, Jia *et al.* [7] propose a method called Compressive Permutation Sampling (CPS), which exploits the sparsity of Shapley values in extensive data. However, the sparsity is inexistent in enterprise federated learning. As a result, CPS cannot work in our scenario. In [8], Jia *et al.* also propose efficient method to calculate Shapley value of data points on machine learning methods relying on K-nearest neighbours, which cannot work in our scenario.

III. Preliminary and Definition

In this section, we first introduce the federated learning framework used in this paper. Then, based on Shapley value [4], we define formally the contribution index of data providers in federated learning tasks.

A. Framework of Federated Learning

Suppose there are n data providers, each with data set $D_i, i \in N = \{1, 2, \cdots, n\}$. We follow the federated learning framework proposed by Google in [2]. The framework is demonstrated in Fig. 1, which is iterative. In each iteration $t \in \{0, 1, \cdots, R-1\}$, there are the following steps:

- Step 1: The server sends a global model $M^{(t)}$ to all the clients (data providers).
- Step 2: Each client, take client i as an example, trains $M^{(t)}$ based on its own data D_i and return the updated sub-model $M_i^{(t)}$ to the server.
- Step 3: The server integrates the sub-models $\{M_i^{(t)} | i \in N\}$ and gets a new global model $M^{(t+1)}$ for the next iteration.

The model integration in Step 3 is as follows. First, the server calculates the gradient of each client by

$$\Delta_i^{(t)} = M_i^{(t)} - M^{(t)}.$$

Then, the server gets the weighted average of all clients' gradients by

$$\Delta^{(t)} = \sum_{i=1}^{n} \frac{|D_i|}{\sum_{i=1}^{n} |D_i|} \cdot \Delta_i^{(t)},$$

where $|D_i|$ is the size of training data D_i. Finally, the server calculates the new global model by the formula of gradient descent,

$$M^{(t+1)} = M^{(t)} + \Delta^{(t)}.$$

B. Definition of Contribution Index

Based on Shapley value [4], we define Contribution Index formally to measure the contribution of the data providers in a federated learning task.

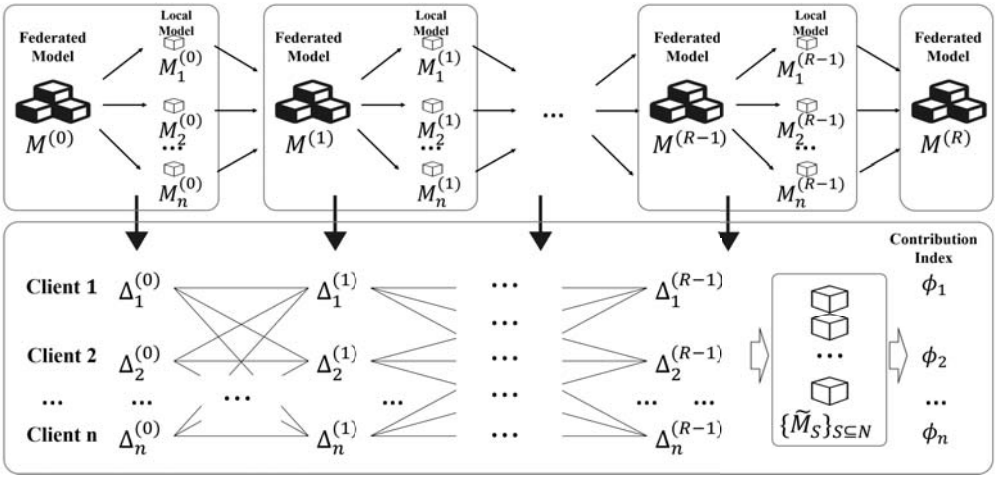

Fig. 2: Schematic diagram for One-Round Model

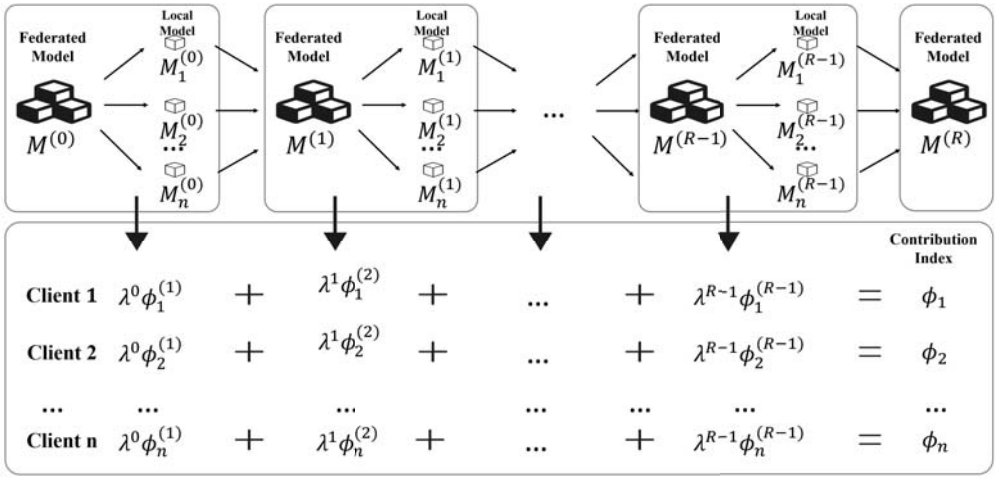

Fig. 3: Schematic diagram for Multi-Rounds Model

Definition 1 (Contribution Index (CI))**.** *There are n data providers with data sets $D_1, D_2, \cdots, D_n$, a machine learning algorithm $\mathcal{A}$ and a standard test set T. We use D_S which is a multi-set to denote $\cup_{i\in S} D_i$ where $S \subseteq N = \{1, 2, \cdots, n\}$. A model trained on D_S through algorithm $\mathcal{A}$ is denoted by $M_S(\mathcal{A})$ and can be abbreviated by M_S if there is no ambiguity. The performance of a model M evaluated on the standard test set T is denoted by $U(M, T)$ and can be abbreviated by $U(M)$ if there is no ambiguity. We use $\phi(\mathcal{A}, D_N, T, D_i)$ to denote the contribution index (CI) of D_i in the context of $D_N, \mathcal{A}$ and T. If there is no ambiguity, $\phi(\mathcal{A}, D_N, T, D_i)$ can be abbreviated as ϕ_i.*

It is expected that the function ϕ should satisfy the following properties.

- If a data set D_j has no effect on the performance of a machine learning algorithm $\mathcal{A}$ on a test set T, the contribution index of D_j should be zero. Formally, if for any subset $S \subseteq N$ we have $U(M_S) = U(M_{S\cup\{j\}})$, then $\phi_i = 0$.
- If two data sets D_i and D_j has the same influence on the performance of a machine algorithm $\mathcal{A}$ on a test set T, they should have the same contribution index. Formally, if for any subset $S \subseteq N\backslash\{i, j\}$ we have $U(M_{S\cup\{i\}}) = U(M_{S\cup\{j\}})$, then $\phi_i = \phi_j$.
- ϕ should have linearity with respect to the test set. Formally, for any two disjoint test set T_1, T_2 and any $i \in N = \{1, 2, \cdots, n\}$, we have $\phi(\mathcal{A}, D_N, T_1 \cup T_2, D_i) = \phi(\mathcal{A}, D_N, T_1, D_i) + \phi(\mathcal{A}, D_N, T_2, D_i)$.

Lemma 1. *Any function ϕ satisfying the above three properties must have the form of*

$$\phi_i = C \sum_{S\subseteq N\backslash\{i\}} \frac{U(M_{S\cup\{i\}}) - U(M_S)}{\binom{n-1}{|S|}}$$

where C is a constant.

Proof. We prove by reducing our problem to a classical cooperative game. In a cooperative game, there are n agents

Algorithm 1: OR; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes :

/* Calculate the Global Model */
$N \leftarrow \{1, 2, \cdots, n\}$;
Initialize $M^{(0)}, \{\widetilde{M}_S^{(0)} | S \subseteq N\}$;
for each round $t \leftarrow 0, 1, 2, ..., R-1$ **do**
  Send $M^{(t)}$ to all the clients;
  $M_i^{(t)} \leftarrow ClientUpdate(i, M^{(t)})$ for client $i \in N$;
  $\Delta_i^{(t)} \leftarrow M_i^{(t)} - M^{(t)}$ for client $i \in N$;
  $M^{(t+1)} \leftarrow M^{(t)} + \sum_{i \in N} \frac{|D_i|}{\sum_{i \in N} |D_i|} \cdot \Delta_i^{(t)}$;
  for each subset $S \subseteq N$ **do**
    $\Delta_S^{(t)} \leftarrow \sum_{i \in S} \frac{|D_i|}{\sum_{i \in S} |D_i|} \cdot \Delta_i^{(t)}$;
    $\widetilde{M}_S^{(t+1)} \leftarrow \widetilde{M}_S^{(t)} + \Delta_S^{(t)}$;
  end for
end for
/* Calculate the CIs */
for i $\leftarrow 1, 2, ..., n$ **do**
  $\phi_i = C \cdot \sum_{S \subseteq N \setminus \{i\}} \frac{U(\widetilde{M}_{S \cup \{i\}}^{(R)}) - U(\widetilde{M}_S^{(R)})}{\binom{n-1}{|S|}}$;
end for
return $M^{(R)}$ and $\phi_1, \phi_2, ..., \phi_n$;

ClientUpdate$(\mathbf{i}, \mathbf{M})$:

$\mathcal{B} \leftarrow$ (split D_i into batches of size B)
for each local epoch $e \leftarrow 1, 2, ..., E$ **do**
  for batch $b \in \mathcal{B}$ **do**
    $M \leftarrow M - \eta \bigtriangledown \mathcal{L}(M; b)$;
  end for
end for
return M to server;

Algorithm 2: MR; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes :

/* Calculate the Global Model */
$N \leftarrow \{1, 2, \cdots, n\}$;
Initialize $M^{(0)}, \{\widetilde{M}_S^{(0)} | S \subseteq N\}$;
for each round $t \leftarrow 0, 1, 2, ..., R-1$ **do**
  /* Calculate the Global Model */
  $M_i^{(t)} \leftarrow ClientUpdate(i, M^{(t)})$ for client $i \in N$;
  $\Delta_i^{(t)} \leftarrow M_i^{(t)} - M^{(t)}$ for client $i \in N$;
  $M^{(t+1)} \leftarrow M^{(t)} + \sum_{i \in N} \frac{|D_i|}{\sum_{i \in N} |D_i|} \cdot \Delta_i^{(t)}$;
  for each subset $S \subseteq N$ **do**
    $\Delta_S^{(t)} \leftarrow \sum_{i \in S} \frac{|D_i|}{\sum_{i \in S} |D_i|} \cdot \Delta_i^{(t)}$;
    $\widetilde{M}_S^{(t+1)} \leftarrow M^{(t)} + \Delta_S^{(t)}$;
  end for
  /* Calculate the round-CIs */
  for i $\leftarrow 1, 2, ..., n$ **do**
    $\phi_i^{(t+1)} = C \cdot \sum_{S \subseteq N \setminus \{i\}} \frac{U(\widetilde{M}_{S \cup \{i\}}^{(t+1)}) - U(\widetilde{M}_S^{(t+1)})}{\binom{n-1}{|S|}}$;
  end for
end for
/* Calculate the CIs */
$\phi_i = \sum_{t=1}^{R} \lambda^t \cdot \frac{\phi_i^{(t)}}{\sum_{i=1}^{n} \phi_i^{(t)}}$ for client $i \in N$
return $M^{(R)}$ and $\phi_1, \phi_2, ..., \phi_n$;

ClientUpdate$(\mathbf{i}, \mathbf{M})$:

$\mathcal{B} \leftarrow$ (split D_i into batches of size B)
for each local epoch $e \leftarrow 1, 2, ..., E$ **do**
  for batch $b \in \mathcal{B}$ **do**
    $M \leftarrow M - \eta \bigtriangledown \mathcal{L}(M; b)$;
  end for
end for
return M to server

and a utility function $U : S \rightarrow \mathbf{R}^+, S \subseteq N$, where N denotes the set of all agents and $U(S)$ measures the utility of a subset of agents S. These agents work together and each of them receives the reward according to their contribution. Lloyd Shapley has proved in [4] that to satisfy all three expected properties above, the only form of the reward of each agent is Shapley value. Given a machine algorithm $\mathcal{A}$, we can view each data provider as an agent and take the performance of the federated model as the utility function. Then the CI of each data provider, measuring everyone's contribution, can be regarded as the reward of each agent in the game. As a result, CI must have a unique form, i.e. the same form as Shapley value. □

IV. Gradient-based Methods

Calculating the CIs according to Lemma 1 directly is time consuming, as models on all the combinations of the data sets need to be trained and evaluated. In this section, we propose two gradient-based methods. The key idea is as follows. We leverage the gradients during the training process of the global model to reconstruct approximately the models trained on different combinations of the data sets. Thus, frequent model retraining is avoided. Based on these reconstructed models, we can evaluate their performance on the standard test set T and estimate the CIs of different data providers.

A. One-Round Reconstruction based Algorithm (OR)

Our first method is called One-Round Reconstruction based Algorithm, or OR in short. The basic idea is shown by Fig. 2 as follows. We gather the gradients information updated by the clients to the server in different training rounds and aggregate gradients corresponding to all the subsets $S \subseteq N = \{1, 2, \cdots, n\}$. With these aggregated gradients, we reconstruct all the models $\{M_S | S \subseteq N\}$ approximately. For example, if we would like to reconstruct the model of $M_{\{1,2\}}$, which means the model trained only on the data sets D_1 and D_2, we just perform weighted averaging according to the data size on the gradients updated from data provider 1 and 2 in each round and use all these gradients to update the initial global model provided by the server. Finally, we calculate according

to Lemma. 1 the CIs of different data providers by evaluating the performance of these reconstructed models.

The details are elaborated by Alg. 1, which has two parts. The first part for the server is shown in lines 1-18. Specifically, in lines 2-3 we use N to denote the set of $\{1, 2, \cdots, n\}$ and initialize a global model $M^{(0)}$ and reconstructed models based on different nonempty subsets $S \subseteq N$ denoted by $\{\widetilde{M}_S^{(0)} | S \subseteq N\}$ using the same randomized model. In lines 4-12, in each training round, the server firstly spreads an initial model $M^{(t)}$ to each client in line 5 and then receives the updated sub-models $\{M_i^{(t)}\}_{i=1,2,\cdots,n}$ from the clients in line 6. In line 7, the gradients of the clients $\{\Delta_i^{(t)}\}_{i=1,2,\cdots,n}$ are calculated for model aggregation. In line, 8 we update the global model maintained by federated learning. In lines 9-11, instead of retraining models on all the nonempty subset of N, we reconstruct these models approximately based on the gradients from the clients. For each $S \subseteq N$ in line 9, we calculate the corresponding gradients by weighted averaging according to the data size in line 10 and using the aggregated gradients to updated the corresponding model in line 11.

In lines 14-17, the CIs of different clients (data providers) are calculated. Specifically, for each client i, we calculate its CI based on Lemma. 1 using the models reconstructed in line 10. In line 18, the trained federated model and the CIs are returned. The second part for the clients is shown in lines 19-25, where clients train their local models based on the model received from the server with their own data, following the classical gradient descent algorithm and report their local models $\{M_i\}_{i=1,2,\cdots,n}$ to the server.

B. Multi-Rounds Reconstruction based Algorithm (MR)

Our second method is called Multi-Rounds Reconstruction based Algorithm, or MR in short. Instead estimating CIs all at once, in MR, we estimate a set of CIs in each training round of federated learning and aggregate them to get the final result.

The basic idea is shown by Fig. 3 as follows. In each round, we gather the gradients updated by the clients to the server. Via these gradients, we reconstruct the models related to different combinations of the data sets by applying the gradients on the global model of the current round. Then we calculate according to Lemma. 1 the CIs of different data providers by evaluating the performance of these reconstructed models. Finally, with these sets of CIs from different training rounds, we perform weighted averaging to get the final results.

The details are elaborated by Alg. 2, which has two parts as well. The first part for the server is shown in lines 1-20. Specifically, lines 1-12 show the calculation of the global model, which is the same with that in Alg. 2. The key difference is embodied in lines 13-19. In lines 13-16, based on Lemma. 1 and our reconstructed models in each round, we can estimate $\phi_i^{(t+1)}$, the round-CI of each client in the t_{th} round. In line 19, we use a parameter $\lambda \in (0, 1)$ to denote the decay factor, which controls the weights of round-CIs in the final result. The idea behind is that with the increment of the iteration rounds, the global model is influenced more and more by all the data sets jointly. Thus, we give higher weights for the earlier rounds. In lines 21-27, the second part for client is the same as the corresponding part in OR.

V. Experiments

We introduce the detailed setup and results of our experiments in this Section.

A. Setup

Data set. The experiments are conducted on the MNIST data set [25]. It contains 60,000+ training images and 10,000+ testing images. Before conducting our experiments, we perform the following pre-processing. Among the training images, the number of images with different labels ranges from 5421 to 6742. We remove some images randomly such that the number of images with different labels is 5421. Thus, the size of the training set is 54210. We do the same on the testing images and finally there are 8920 images in the test set with each label having the same number of images.

In reality, the data sets owned by different data providers may have different size, distribution and quality. To verify the performance of our methods on different data settings, we further processing our training set into 5 cases as follows.

- **The same distribution with the same data size**. In this case, we split the data set into five parts with the same size randomly. We ensure that the numbers of images with the same label is mutually equal among the five parts.
- **Different distributions with the same data size**. In this case, we split the data set into five parts with the same size. However, the distribution of the labels of the five parts are different. Specifically, we let data provider 1 have 40% of label '0', 40% of label '1' and the data points with the remaining eight labels share the remaining 20% of data provider 1's data. The rest can be done in the same manner.
- **The same distribution with different data sizes**. In this case, we split the data set into five parts randomly and the ratio of data size is 2:3:4:5:6. We ensure that each part have the same number of images with the different labels.
- **Noised data on labels with the same data size**. In this case, we split the data set into five parts with the same size randomly. We also ensure that the numbers of images with the same label is mutually equal among the five parts. We then change the labels of different data providers randomly by $0\%, 5\%, 10\%, 15\%, 20\%$.
- **Noised data on features with the same data size**. In this case, we split the data set into five parts with the same size randomly. For each part, we generate 0x-4x Gaussian noise, where 0x is for the feature without noise. After that we add different levels of noise to all the features of different parts respectively. Fig. 4 is an example where we pick one feature from MNIST data and add different levels of noise to show the influence of feature noise.

Environment. The experiments are conducted on a machine with Intel(R) Xeon(R) Platinum 8163 CPU @ 2.50GHz and

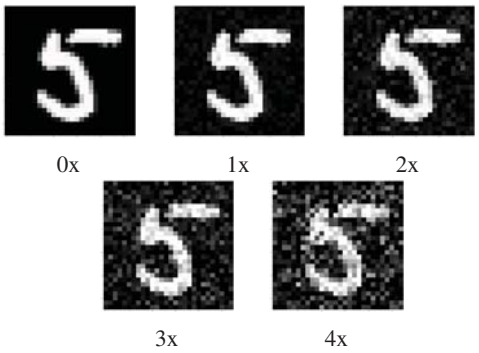

Fig. 4: Noised examples of MNIST

32GB main memory. The codes are implemented on python 3.6 with TensorFlow [23] 1.14.0 and TensorFlow Federated [24] 0.8.0.

Compared Algorithms. The following algorithms are compared.

- **Exact.** This method calculates the CIs of the data providers according to Lemma. 1. Specifically, it trains federated models based on different combinations of the data sets and these models are evaluated on the standard test set.
- **Extended-TMC-Shapley.** This method is extended from the Algorithm 1 Truncated Monte Carlo Shapley (denoted by TMC-Shapley in short) of [6]. Extended-TMC-Shapley first samples a random permutation π of all $n!$ permutations of $1, 2, \cdots, n$ which is denoted by Π. Then it train models according to the permutation π and calculates the CI of each data provider according to

  $$\phi_i = \mathbb{E}_{\pi\sim\Pi}[U(M_{\pi_i}) - U(M_{\pi_{i-1}})].$$

  where π_i means the set of the first i numbers in π and thus M_{π_i} means the model trained on the first i data sets in the permutation π. In the implementation, we record the results of different combinations, and once we need to evaluate the performance under some recorded combination, we can directly use the corresponding results.
- **Extended-GTB.** This method is extended from the Algorithm 1 Group Testing Based SV Estimation of [7]. It samples some subsets S of $N = \{1, 2, \cdots, n\}$ and trains models M_S. Based on the performance of these models, it solves a feasibility problem to estimate the CIs of each data provider under some constraints. In [7], this method is mainly stated as a theoretical result, which can achieve comparative performance with less training round. In our experiments, we set the same number of model training for Extended-GTB and Extended-TMC-Shapley. In the implementation, the results of models trained on different combinations of the datasets are recorded for reuse as well. The feasibility problem is solved by Mathematica 11.2 [26]. If the feasibility problem has no solutions, we relax the constraints until it has one.
- **One-Round.** The algorithm is proposed in Sec. IV-A and demonstrated by Alg. 1.
- **Multi-Round.** The algorithm is proposed in Sec. IV-B and demonstrated by Alg. 2.
  In the experiment, we find that at the very beginning of federated learning, the global model may perform poorly as the model is initialized randomly. As a result, some data providers may have negative CIs. Thus, in the first several rounds of training, once the CI of someone is negative, we adopt the egalitarianism and assign each data provider the same CI.

Evaluation Metrics. We compare different methods on the following metrics.

- **Time.** The total time of model training and the calculation of contribution index is compared.
- **Cosine Distance.** Let the vectors of *normalized* contribution index of different data providers calculated according to the definition (Lemma. 1) and by an approximated method is denoted by $\phi^* =< \phi_1^*, \phi_2^*, \cdots, \phi_n^* >$ and $\phi =< \phi_1, \phi_2, \cdots, \phi_n >$, respectively. The Cosine Distance is defined by

  $$D_C = 1 - cos(\phi^*, \phi) = 1 - \frac{\sum_{i=1}^n \phi_i^* \times \phi_i}{\sqrt{\sum_{i=1}^n {\phi_i^*}^2} \times \sqrt{\sum_{i=1}^n {\phi_i}^2}}.$$

- **Euclidean Distance.** The Euclidean Distance is defined by

  $$D_E = \sqrt{\sum_{i=1}^n (\phi_i^* - \phi_i)^2}.$$

- **Maximum Difference.** The Maximum Difference is defined by

  $$D_M = \max_{i=1}^n |\phi_i^* - \phi_i|.$$

  This metric is used to measure the maximum difference of percentage that a data provider should be allocated by the definition and by approximated calculation.

B. Results

We next analyze the experimental results on the five cases of data settings.

Results on the same distribution with the same data size. On this setting, each data provider has the same quality and quantity of data. Trivially, the expected result is that each data provider has the same contribution index. We first study the time cost in Fig.5 (a). Each column presents the result of different methods. We can observe that OR has the highest efficiency. The time cost of MR and Extended-TMC-Shapley, twice as OR, are also acceptable. MR is more efficient than Extended-TMC-Shapley. However, with frequent model retraining, the time cost of Extended-GTB and Exact is 5x-10x as OR and MR. Next, we study the the cosine distance between the exact CIs and CIs calculated by other methods. In Fig.5 (b), we can observe that the distances of OR and MR are nearly to 0, i.e. the CIs calculated via OR and MR are almost the

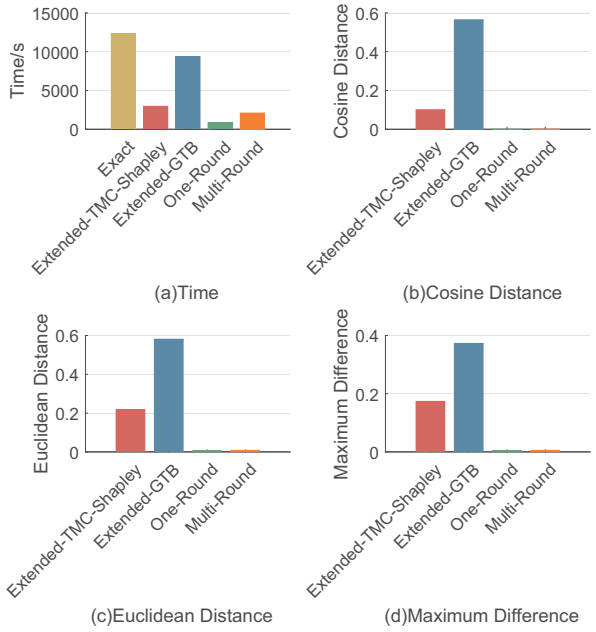

Fig. 5: The same distribution with the same data size

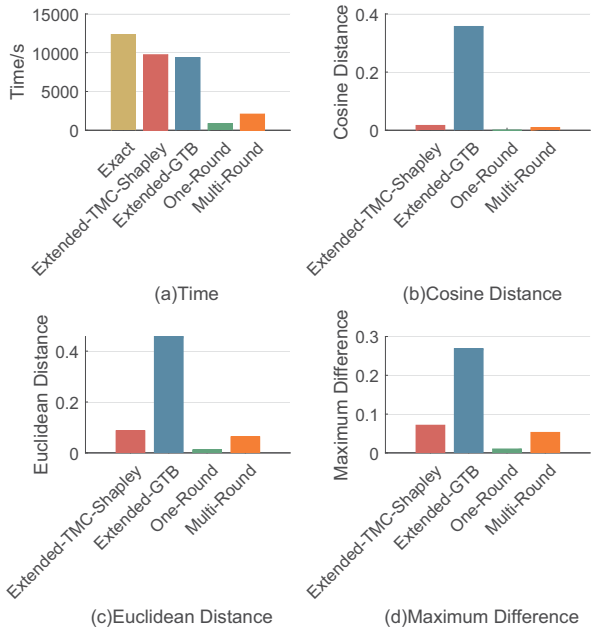

Fig. 6: Different distribution with the same data size

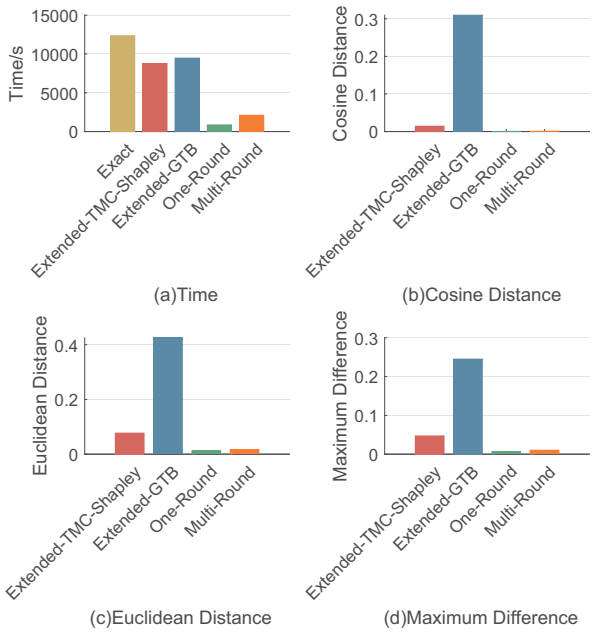

Fig. 7: The same distribution with different data size

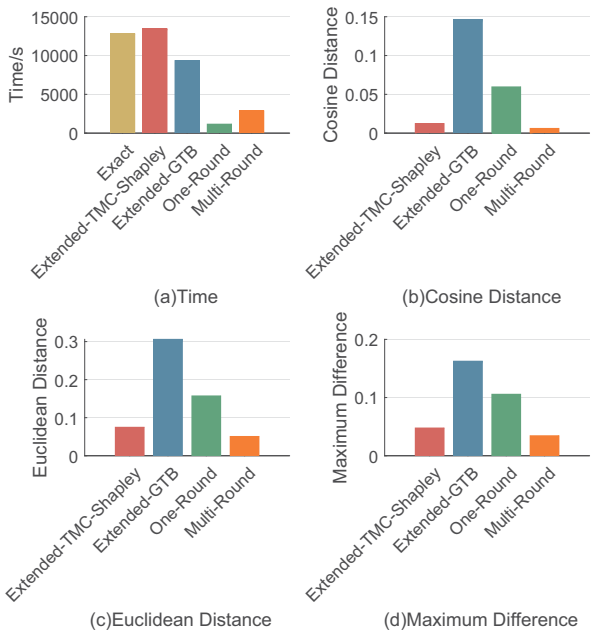

Fig. 8: Noised data on labels with the same data size

same with the exact CIs. the effects of our methods are almost same as the exact CIs, while there are obvious difference between the two baseline algorithms (Extended-TMC-Shapley and Extended-GTB) and the exact CIs. In fig.5 (c) and (d), we can observe that, on the metrics of both euclidean distance and maximum difference, OR and MR perform 30x-40x better than TMC-Shapley. The Extended-GTB is very poor in accuracy.

Results on different distributions with the same data size. We next study the results on different distribution with the same data size. In Fig. 6 (a), we can observe that, on time cost, OR still ranks first and MR still ranks second. However, Extended-TMC-Shapley cannot maintain its performance in this setting. The similar time cost of Extended-TMC-Shapley and Extended-GTB are about 10x higher than OR and about 4x higher than MR. We then introduce the results of metrics in Fig. 6 (b), (c) and (d). Firstly, OR performs better than

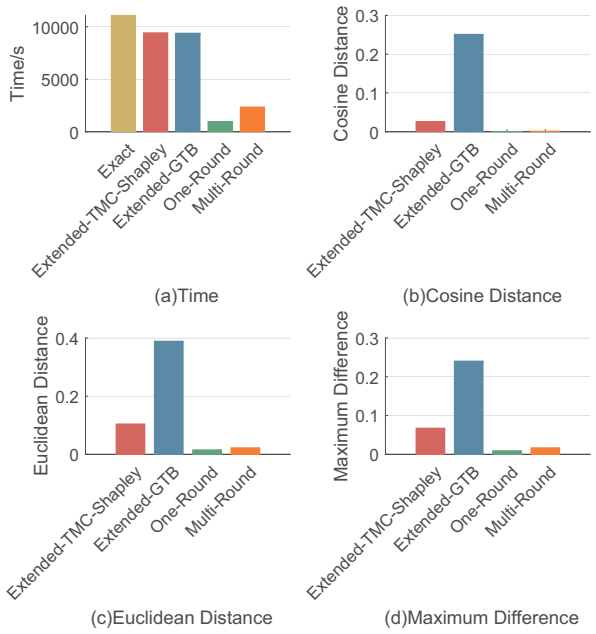

Fig. 9: Noised data on features with the same data size

other algorithms in all metrics. We can almost regard the CIs calculated by OR as the exact one. MR performs better than Extended-TMC-Shapley and Extended-GTB on all three metrics for accuracy with much lower time cost. As a result, OR and MR also dominate the two baseline algorithms in this setting.

Results on the same distribution with different data sizes. In this setting, we keep the distribution of features the same and vary the data size of different providers. In Fig. 7 (a), OR and MR still have much lower time cost than other algorithms. On cosine distance in Fig. 7 (b), we can observe that OR and MR approximate the exact CIs very well and dominate the baseline algorithms. Extended-GTB is the worst. In Fig. 7 (c) and (d), the results of OR and MR are similar and dominant. Extended-TMC-Shapley is less accurate while Extended-GTB's performance is unsatisfactory.

Results on noised labels with the same data size. We next study the influence of noised label in Fig. 8. On time cost, the results in Fig. 8 (a) are similar with that in other settings. OR and MR are the most efficient than the other algorithms. In Fig. 8 (b), we can observe that OR's performance decline obliviously. MR's performance is stable. It performs 2x better than Extended-TMC-Shapley, 10x better than OR, and 20x better than Extended-GTB. In Fig. 8 (c) and Fig. 8 (d), MR also dominates other algorithms. OR is better than Extended-GTB but worse than Extended-TMC-Shapley. In this setting, OR dose not perform well as above and seems to be sensitive to the noise on label. Meanwhile,, MR shows its robust and dominates other approximation algorithms under the influence of noised label.

Results on noised features with the same data size. At last, we study the results under noised features. The results are shown in Fig. 9. Overall, OR is slightly better than MR both on time cost and accuracy and OR and MR are obviously better than other baseline algorithms. In Fig. 9 (b), OR and MR perform 20x better than Extended-TMC-Shapley and 200x better than Extended-GTB on cosine distance metric. In Fig. 9 (c), OR and MR perform 4.6x better than Extended-TMC-Shapley and 17x better than Extended-GTB on euclidean distance metric. In Fig. 9(d) we can observe that the maximum difference of OR, MR, Extended-TMC-Shapley and Extended-GTB are 0.9%, 1.7%, 6.7% and 24% respectively. As a result, OR and MR dominate the two baseline algorithms in this setting.

C. Summary

Based on the above experiments, it can be summarized that

- **Time.** Extended-TMC-Shapley and Extended-GTB are the most time consuming. The two baseline algorithms are similar in efficiency. The cost time of OR and MR are robust to different settings and they are the most efficient. OR is better than MR.
- **Performance.** MR's performance is stable and satisfactory in all the settings. OR approximates the exact CIs best in most settings. However, it is sensitive to the noise on the label. Extended-TMC-Shapley and Extended-GTB perform the worst. Here we make a brief discussion on the performance of Extended-GTB. During solving the feasible problem in Extended-GTB, we find that some inaccurate constraints may make the problem unsolvable. To find an instance of solution, we have to repeat relaxing the constraints of all inequalities, which leads to the CIs inaccurate. Thus, specially designed methods to solve the feasible problem in Extended-GTB are necessary.

VI. Conclusion

In this paper, we define a concept called Contribution Index based on Shapley value to measure the contribution of different data providers in a horizontal federated learning task. Based on Contribution Index, the profit earned by the joint model can be allocated reasonably and thus data owners can be attracted to join in data federation. The calculation of Contribution Index according to the definition is time consuming, as a large number of model training on different combinations of all the data sets need to be performed. We propose two gradient-based methods, whose key idea is to reconstruct models through the intermediate results of the federated learning process. We conduct extensive experiments on the MNIST data set and the results demonstrate that our methods can approximate the exact Contribution Index by definition effectively and speed up by 2x-100x compared with other baselines.

Acknowledgment

We are grateful to anonymous reviewers for their constructive comments. This work is partially supported by the National Science Foundation of China (NSFC) under Grant No. 61822201 and U1811463. Yongxin Tong is the corresponding author of this paper.

REFERENCES

[1] European Parliament and The Council of the European Union, "The general data protection regulation (GDPR)," in *https://eugdpr.org*, 2016.

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS, Fort Lauderdale, USA, 20-22 April 2017*, pp. 1273–1282.

[3] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, p. 12, 2019.

[4] L. S. Shapley, "A value for n-person games," *Annals of Mathematical Studies*, vol. 28, pp. 307–317, 1953.

[5] R. B. Myerson, *Game theory*. Harvard university press, 2013.

[6] A. Ghorbani and J. Zou, "Data shapley: Equitable valuation of data for machine learning," pp. 2242–2251, 2019.

[7] R. Jia, D. Dao, B. Wang, F. A. Hubis, N. Hynes, N. M. Gürel, B. Li, C. Zhang, D. Song, and C. J. Spanos, "Towards efficient data valuation based on the shapley value," in *The 22nd International Conference on Artificial Intelligence and Statistics*, 2019, pp. 1167–1176.

[8] R. Jia, D. Dao, B. Wang, F. A. Hubis, N. M. Gurel, B. Li, C. Zhang, C. Spanos, and D. Song, "Efficient task-specific data valuation for nearest neighbor algorithms," *Proceedings of the VLDB Endowment*, vol. 12, no. 11, pp. 1610–1623, 2019.

[9] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, 2017, pp. 1175–1191. [Online]. Available: https://doi.org/10.1145/3133956.3133982

[10] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, 2017, pp. 19–38. [Online]. Available: https://doi.org/10.1109/SP.2017.12

[11] P. Mohassel and P. Rindal, "Aby^3: A mixed protocol framework for machine learning," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, 2018, pp. 35–52. [Online]. Available: https://doi.org/10.1145/3243734.3243760

[12] S. Hardy, W. Henecka, H. Ivey-Law, R. Nock, G. Patrini, G. Smith, and B. Thorne, "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption," *CoRR*, vol. abs/1711.10677, 2017. [Online]. Available: http://arxiv.org/abs/1711.10677

[13] I. Giacomelli, S. Jha, M. Joye, C. D. Page, and K. Yoon, "Privacy-preserving ridge regression with only linearly-homomorphic encryption," in *Applied Cryptography and Network Security - 16th International Conference, ACNS 2018, Leuven, Belgium, July 2-4, 2018, Proceedings*, 2018, pp. 243–261. [Online]. Available: https://doi.org/10.1007/978-3-319-93387-0_13

[14] M. Abadi, A. Chu, I. J. 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, Vienna, Austria, October 24-28, 2016*, 2016, pp. 308–318. [Online]. Available: https://doi.org/10.1145/2976749.2978318

[15] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *CoRR*, vol. abs/1712.07557, 2017. [Online]. Available: http://arxiv.org/abs/1712.07557

[16] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018. [Online]. Available: https://openreview.net/forum?id=BJ0hF1Z0b

[17] E. Kharitonov, "Federated online learning to rank with evolution strategies," in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM 2019, Melbourne, VIC, Australia, February 11-15, 2019*, 2019, pp. 249–257. [Online]. Available: https://doi.org/10.1145/3289600.3290968

[18] J. Konecný, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *CoRR*, vol. abs/1610.02527, 2016. [Online]. Available: http://arxiv.org/abs/1610.02527

[19] S. Ramaswamy, R. Mathews, K. Rao, and F. Beaufays, "Federated learning for emoji prediction in a mobile keyboard," *CoRR*, vol. abs/1906.04329, 2019. [Online]. Available: http://arxiv.org/abs/1906.04329

[20] M. Yurochkin, M. Agarwal, S. Ghosh, K. H. Greenewald, T. N. Hoang, and Y. Khazaeni, "Bayesian nonparametric federated learning of neural networks," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, 2019, pp. 7252–7261. [Online]. Available: http://proceedings.mlr.press/v97/yurochkin19a.html

[21] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, 2019, pp. 4615–4625. [Online]. Available: http://proceedings.mlr.press/v97/mohri19a.html

[22] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecný, S. Mazzocchi, H. B. McMahan, T. V. Overveldt, D. Petrou, D. Ramage, and J. Roselander, "Towards federated learning at scale: System design," *CoRR*, vol. abs/1902.01046, 2019. [Online]. Available: http://arxiv.org/abs/1902.01046

[23] M. Abadi, A. Agarwal, P. Barham, and et al, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[24] "Tensorflow federated," in *www.tensorflow.org/federated/federated_learning*.

[25] Y. LeCun, C. Cortes, and C. J. Burges, "The MNIST Database," in *http://yann.lecun.com/exdb/mnist/*.

[26] W. R. Inc., "Mathematica, version 11.2," champaign, IL, 2017.