Efficient Participant Contribution Evaluation for Horizontal and Vertical Federated Learning

Junhao Wang, Lan Zhang*, Anran Li, Xuanke You, Haoran Cheng
School of Computer Science and Technology, University of Science and Technology of China, Hefei, China
{junhaow,anranLi,yxkyong,chr990315}@mail.ustc.edu.cn, {zhanglan}@ustc.edu.cn

2022 IEEE 38th International Conference on Data Engineering (ICDE) | 978-1-6654-0883-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICDE53745.2022.00073

***Abstract*—Federated Learning (FL) enables multiple participants to collaboratively train a model in a privacy-preserving way. The performance of the FL model heavily depends on the quality of participants' local data, which makes measuring the contributions of participants an essential task for various purposes, e.g., participant selection and reward allocation. The Shapley value is widely adopted by previous work for contribution assessment, which, however, requires repeatedly leave-one-out retraining and thus incurs the prohibitive cost for FL. In this paper, we propose a highly efficient approach, named DIG-FL, to estimate the Shapley value of each participant without any model retraining. It's worth noting that our approach is applicable to both vertical federated learning (VFL) and horizontal federated learning (HFL), and we provide concrete design for VFL and HFL. In addition, we propose a DIG-FL based reweight mechanism to improve the model training in terms of accuracy and convergence speed by dynamically adjusting the weights of participants according to their per-epoch contributions, and theoretically analyze the convergence speed. Our extensive evaluations on 14 public datasets show that the estimated Shapley value is very close to the actual Shapley value with Pearson's correlation coefficient up to 0.987, while the cost is orders of magnitude smaller than state-of-the-art methods. When there are more than 80% participants holding low-quality data, by dynamically adjusting the weights, DIG-FL can effectively accelerate the convergence and improve the model accuracy.**

I. Introduction

The rapid development of Artificial Intelligence, and social networking applications is incurring enormous growth of the data generated at the network edge, which makes data privacy concerns particularly important [1], [2]. Federated learning (FL) is an emerging technology that allows multiple participants to collectively train a global machine learning model without exposing their local training data and training process. Based on how data is distributed among the participants in the feature and sample ID space, there are two widely adopted federated learning frameworks: vertical federated learning (VFL) and horizontal federated learning (HFL). The architectures of VFL and HFL systems are quite different by design. HFL applies to scenarios that participants share the same feature space, but have different samples, where the global model is obtained by aggregating local parameters of participants [3], [4], [5]. VFL applies to scenarios that participants share the same sample ID space but have different data features. VFL builds a global model by computing gradients with features from participants in a privacy-preserving manner [3], [6], [7], [8], e.g., using encryption techniques.

In FL, the performance of the global model largely depends on the quality of local data. For example, when many participants possess non-IID (Independent and Identically Distributed) or erroneous data, it hinders the global model from achieving a good performance [9], [10], [11]. Therefore, it is essential to identify participants holding low-quality data, which is, however, intractable due to the invisibility of participants' local data. This challenge drives us to explore an effective way to measure the contributions of participants in FL systems, which could bring multi-fold benefits: (1) It helps us to understand behaviors of FL models by tracing back to distributed training datasets; (2) It can localize low-quality participants and thus reduce their impact to mitigate performance degradation or avoid adversarial sample attacks; (3) During the training process, weights of participants can be adjusted according to their contributions so as to boost the model convergence; (4) For the commercial use of FL, fair credit/reward allocation for participants based on their contributions is needed.

Many efforts have been devoted to measuring contributions of training samples and participants in machine learning systems. For centralized machine learning, a series of approaches aim to interpret the model behavior by analyzing the influence of data samples on the model's predictions [12], [13], [14]. However, they require access to training data, thus cannot be directly adopted by FL. Recently, Xue et al. [15] and Li et al. [16], [17] use the influence function-based method to measure the influence of participants in FL, which requires participants to calculate and upload their Hessian matrices or part of Hessian matrices. Applying those methods to calculate contribution requires exponential calculation of influence function, resulting in large extra overhead. Zhang et al. [18] simply use cosine distance between each participant's gradient and the gradient of the final global model to measure participant contribution for HFL. All those methods do not use the Shapley value [19], hence they do not have the efficiency, symmetry, linearity, and null player properties. Moreover, they are only applicable to HFL, not to VFL. There are some prior arts leveraging the Shapley value to measure contributions of samples [20], [21] for centralized learning. However, applying those methods to FL requires repeatedly retraining the model, which imposes unacceptable computation and communication overhead, especially for resource-constrained devices[22]. Few

*Lan Zhang is the corresponding author.

2375-026X/22/$31.00 ©2022 IEEE
DOI 10.1109/ICDE53745.2022.00073

work use the Shapley value to assess the contributions of participants in FL [23], [24]. For HFL, Song et al. [23] propose two methods to approximate the Shapley value without retraining model, which, however, still require to exponentially test model performance, and thus impose prohibitive computation cost. For VFL, Wang et al. [24] propose a method to estimate the Shapley value for an individual feature, which, however, introduces extra severe privacy risk since it needs to access and permutate the training data.

In this work, we aim to propose a highly efficient approach to accurately assess the Shapley value based contribution of each participant for both VFL and HFL. Obtaining the contribution assessment, we can design fair incentive mechanisms, localize low-quality participants or reweight participants to improve the model performance and convergence speed. Towards this ambitious goal, we need to answer the following challenging questions:

(1) How to accurately measure the contribution of each participant with minimal extra cost, which should be significantly smaller than the training cost? The Shapley value provides a principled way, which is characterized by a collection of desirable properties, to evaluate how important each participant is to the overall collaborative learning. Following the definition of the Shapley value, existing methods require exponentially retraining the model or testing model performance, which is prohibitively expensive for FL. Therefore, a desired approach should assess contributions to closely approximate the actual Shapley values with minimal extra cost.

(2) How to design a general approach applicable to both VFL and HFL, especially for VFL using different cryptography techniques? HFL and VFL have completely different architectures, and both of them have various frameworks across a wide range of applications [3], [6], [8]. It is non-trivial to design an approach applicable to various FL frameworks.

(3) How to respect participants' data privacy when assessing their contributions? The invisibility of participants' local data is the most attractive feature of FL. Therefore, we need to measure the impact of each participant in the training process without access to their local data. It is challenging, since exiting contribution measuring methods often require extra computation and transmission, which cause privacy risk.

To address these questions, we use Shapley value to measure the contribution of each participant, and first define the utility function as the change in loss on the validation dataset via leave-one-out model retraining. Then, we theoretically analyze the impact of each participant on global gradients and the utility function, and prove that the change in utility function caused by removing each participant satisfies additivity. Leveraging the additivity, we turn the exponential calculation of the Shapley value into the linear calculation. Moreover, we utilize the training logs to calculate an approximation of the actual Shapley value, which requires no extra model training or access to local data. Our proposed approach, named DIG-FL, can efficiently and accurately measure the contributions of participants for HFL and VFL. And protecting data privacy and security is a major issue for artificial intelligence applications [25], [26], [27]. We give the definition of privacy and analyze the privacy risks that our algorithm may have.

Our contributions are summarized as follows:

•We propose a novel approach DIG-FL to efficiently measure the Shapley value based contributions of participants for both HFL and VFL. We theoretically show that DIG-FL can accurately approximate the actual Shapley value using only training logs, which requires no extra model training or access to local data. We turn the exponential calculation of the Shapley value into the linear calculation, thus DIG-FL costs significantly smaller computation and communication cost than that of previous approaches. Besides, two of our algorithms for HFL and VFL do not introduce any additional privacy risk to FL systems.

•Based on the per-epoch contribution measured by DIG-FL, we design a participant reweight mechanism to improve the model training in terms of accuracy and convergence speed by dynamically adjusting the weights of participants. For both HFL and VFL, we theoretically analyze the convergence speed by using our reweight mechanism.

•We extensively evaluated our approach for HFL and VFL on 14 public datasets. For HFL, the evaluation results show that the estimated and the actual Shapley values are highly correlated, their Pearson's correlation coefficient (PCC) is 0.968 on MNIST, 0.935 on CIFAR10, 0.952 on MOTOR and 0.833 on REAL. For VFL, we adopted a well-known FL framework [3], [28], as an example to test our approach in VFL systems. The estimated Shapley value and actual one match closely, with 0.987 PCC for vertical linear regression and 0.940 PCC for vertical logistic regression. Compared with conventional methods calculating the actual Shapley value, DIG-FL reduces computation cost by orders of magnitude, e.g., from 8.9×10^5 s to 1.1×10^3 s on MNIST for HFL, and from 76,584.7s to 13.77s on Seoul Bike Sharing Demand dataset [29] for VFL. Compared with state-of-the-art estimation methods, DIG-FL achieves better accuracy and consumes several orders of magnitude less cost. When there are more than 80% participants holding low-quality data, our reweight mechanism can effectively accelerate the convergence and improve the model accuracy from 67.7% to 95.3% on MNIST, from 70.9% to 89.9% on CIFAR10, from 55.2% to 86.5% on MOTOR and from 47.4% to 77.1% on REAL.

II. Problem and Main Idea

A. Problem Description

In this work, our goal is to design an approach to efficiently and accurately assess the contribution of each participant for both HFL and VFL. We adopt the Shapley value as the metric due to its appealing properties and aim to accurately estimate the Shapley value with very constrained cost.

In HFL, each participant updates a local model with local training data, and a parameter server aggregates updates from all participants to train a global model. In VFL, there is usually a trusted third-party generating encryption key pairs. Each participant owns some features and a part of the complete

model. All participants collaboratively train the complete model utilizing multiparty computation.

Before presenting our idea, we first provide a unified formalization of HFL and VFL. Given n participants $\mathcal{C} = \{1, 2, ..., n\}$ and a server, model training starting from epoch 1 to epoch τ, in epoch t, participant i sends local update $\delta_{t,i}$ to the server. The server collects $\Lambda_t = \{\delta_{t,1}, \delta_{t,2}, ..., \delta_{t,n}\}$, calculates the global gradient $\mathcal{G}_t$, and sends it to all participants. Each participant computes local update based on $\mathcal{G}_t$. This process continues to iterate until the model converges. For simplicity, we focus on the model training process and ignore the encryption details of VFL for now, and present the detailed VFL protocol in Sec.IV. We assume that at least one participant's local data meet the quality standard of the learning task and the server holds a high-quality (e.g., error-free and IID) validation dataset $\mathcal{D}^v$ to measure the performance of the global model. Note that, the volume of $\mathcal{D}^v$ is usually much smaller than that of the training data, therefore $\mathcal{D}^v$ is insufficient for model training but is easy to collect.

Since privacy preservation is the most attractive feature of FL, our design should not introduce any extra privacy risks during measuring contributions. Specifically, for HFL, we assume semi-honest server and participants, which is a common setting in most HFL work ([3], [30] et al.). Here we define two levels of privacy: 1) level-1: no participant's local training data is transmitted/exposed to any other parties; 2) level-2: except for the local model, which is necessary for HFL model training, no additional information is transmitted/exposed to any other parties. Both privacy definitions do not allow direct exposure of local training data. Level-2 privacy is more stringent than level-1 privacy, because level-1 privacy allows the transmission of intermediate computation results (e.g., Hessian matrix) other than the local model, but level-2 privacy does not. Conventional HFL systems usually meet the level-2 privacy definition. Sometimes, to achieve a higher privacy protection level, HFL can adopt techniques such as homomorphic encryption [31], differential privacy [32] or secret sharing [33] to mask local model. Those techniques can also be adopted by DIG-FL, but how to apply them is out of scope of this work. For VFL, as previous work ([6], [33], [34] et al.), we assume a trusted third-party responsible for key management and semi-honest participants. The privacy definition is that any party can learn nothing from other parties beyond what is revealed by his/her input and output.

B. Main Idea

The Shapley value[19] is a broadly adopted solution concept in cooperative game theory to measure how each participant contributes to the overall cooperation. Given a coalitional game (V, N), where N is a set of n participants and $V(\cdot)$ is a utility function defined as $V : 2^N \to \mathbb{R}$, the Shapley value of participant i is:

$$\phi_i(V) = \sum_{S \subseteq N} \frac{|S|!(n - |S| - 1)!}{n!}(V(S) - V(S\backslash\{i\})). \quad (1)$$

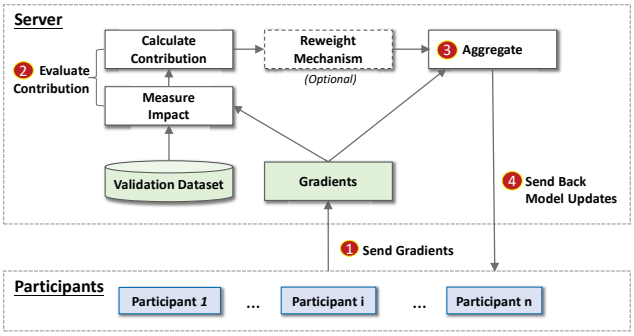


Fig. 1. **System overview of DIG-FL in HFL. The reweight mechanism step is optional, which can be performed when we aim to identify the negatively influential participants and adjust the weights to boost model convergence based on per-epoch contributions.**

Here, $V(\cdot)$ is the untility function, $V(S)$ is the worth of coalition S, indicating the total expected surplus the member of S can obtain by cooperation. The Shapley value can be interpreted as: when participant i joins a coalition S, he/she demands his/her marginal contribution $V(S) - V(S\backslash\{i\})$ as a fair payoff, and then for each participant the payoff is the average of his/her marginal contribution over all possible coalitions.

Generally, the utility function is defined as the performance of the global model, that is, the loss of the global model on the validation dataset. The utility function of a coalition S is defined as:

$$V(S) = loss^v(\theta(\varnothing)) - loss^v(\theta_\tau(S)), \quad (2)$$

where $loss^v$ is the loss function on the validation dataset, $\theta(\varnothing) = \theta_0$ is the initial global model and $\theta_\tau(S)$ is the final global model trained by the coalition S. The marginal contribution of participant z joining in the coalition $S \setminus \{z\}$ is $V(S) - V(S \setminus \{z\})$.

Directly applying Shapley value to the FL system requires the exponential calculation of the utility function change and marginal contribution. For example, to calculate the marginal contribution of participant i joining coalition S, the model needs to be trained twice, once with i and once without i. The calculation of the Shapley value requires iteratively calculating the marginal contribution. Some existing works ([20], [21] et al.) focus on reducing the number of repetitive training to reduce the cost in centralized learning. However, applying those methods to FL still requires repeatedly retraining the model, which imposes unacceptable computation and communication overhead, especially for resource-constrained devices [22]. Therefore, we aim to efficiently calculate Shapley value in a variety of FL systems, without model retraining.

The most critical question in our design is how to measure the changes in global model performance caused by participant(s) withdrawing from the FL system, without model retraining. Since the change of model performance essentially comes from the change of model parameters, we translate

this question into how to measure the changes in global model's parameters and gradients caused by participant(s) withdrawing, which is the impact of participant(s), without model retraining. To answer this question, we delve into the training process of FL and find the way to efficiently measure the impact of each participant. Then we model the utility function based on impacts of participants and propose to use only the training log (local gradients from all participants) to estimate the marginal contribution. DIG-FL uses the training log and the validation dataset to calculate per-epoch contributions and aggregate them to approximate the actual Shapley value during the whole training process. The per-epoch contribution can be utilized for various purposes, such as dynamically reweighting participant, selecting optimal participant under a budget constraint, a fairer contribution-based payment, etc. In this work, we design a participant reweight mechanism to improve the model training in terms of accuracy and convergence speed.

Taking HFL as an example, Fig. 1 shows the overview of DIG-FL working in an HFL system. There are four main steps: 1) Participants update model using local training data and send local gradients to the server; 2) The server evaluates the contribution of each participant in each epoch, including impact measurement (Sec.II-C) and contribution calculation (Sec.II-D) ; 3) The server performs gradient aggregation to obtain the updated global model; 4) The server sends model updates to all participants. The reweight mechanism is optional, which can be enabled when we want to mitigate the negative influence of participants with low-quality training data. We will introduce the details of the reweight mechanism in Sec.II-F.

C. Impact Measurement

We study the change in global model gradients and parameters due to removing a participant or a subset of participants in the training process for HFL and VFL.

1) For HFL: In epoch t, participant i updates the current global model θ_{t-1} using local data to obtain the local model $\theta_{t-1,i}$ and sends it to the server. The server aggregates local models from participants to obtain the global model $\theta_t = \frac{1}{n}\sum_{i=1}^{n}\theta_{t-1,i}$. Let the local update of participant i be $\delta_{t,i} = \theta_{t-1} - \theta_{t-1,i}$. The server gets $\Lambda_t = \{\delta_{t,1}, \delta_{t,2}, ..., \delta_{t,n}\}$, computes the global gradient $\mathcal{G}_t = \frac{1}{n}\sum_{i=1}^{n}\delta_{t,i}$ and update the global model $\theta_t = \theta_{t-1} - \mathcal{G}_t$. When participant z is removed in training process, at epoch t, the gradients $\mathcal{G}_t^{-z}$, change in gradients is $\Delta\mathcal{G}_t^{-z} = \mathcal{G}_t^{-z} - \mathcal{G}_t$ and change in parameters is $\Delta\theta_t^{-z} = \theta_t^{-z} - \theta_t = -\sum_{j=1}^{t}\Delta\mathcal{G}_j^{-z}$.

Lemma 1. *For HFL, in epoch t, if the loss function is twice-differentiable, when participant z is removed, change in gradient is:*

$$\Delta\mathcal{G}_t^{-z} = \mathcal{G}_t^{-z} - \mathcal{G}_t = -\frac{1}{n}\delta_{t,z} + \alpha_t\Omega_t^{-z}, \tag{3}$$

where $\Omega_t^{-z} = H_{\theta_{t-1}}(\sum_{j=1}^{t-1}\Delta\mathcal{G}_j{}^{-z})$ and α_t is the learning rate at epoch t.

And the change satisfies additivity, that is, when a subset of participants S is removed, change in gradients is $\Delta\mathcal{G}_t^{-S} = \sum_{i\in S}\Delta\mathcal{G}_t^{-i}$.

Proof Sketch. *We first study the change of model parameters and gradients after upweighting participant z by a small ε. We assume that the initialization model is θ_0. For HFL, at epoch t, the global gradient is*

$$\mathcal{G}_t = \frac{1}{n}\sum\delta_{t,i} = \alpha_t\frac{1}{n}\sum\nabla loss(i,\theta_{t-1}), \tag{4}$$

where $\nabla loss(i,\theta_{t-1})$ is the local gradient calculated by participant i using the local data and global model θ_{t-1} of the last epoch.

If upweight participant z by ε during the whole training, at epoch t, the global gradient is

$$\begin{aligned}\mathcal{G}_t^{\varepsilon z} &= \frac{1}{n}\sum\delta_{t,i}^{\varepsilon z} + \varepsilon\delta_{t,i}^{\varepsilon z}\\ &= \alpha_t\frac{1}{n}\sum\nabla loss(i,\theta_{t-1}^{\varepsilon z}) + \alpha_t\varepsilon\nabla loss(z,\theta_{t-1}^{\varepsilon z}),\end{aligned} \tag{5}$$

where $\theta^{\varepsilon z}$ is the global model after upweighting z by ε.

At epoch t, the change of global gradients is:

$$\begin{aligned}\Delta\mathcal{G}_t^{\varepsilon z} &= \mathcal{G}_t^{\varepsilon z} - \mathcal{G}_t\\ &= \alpha\varepsilon\nabla loss(z,\theta_{t-1}) + \alpha H_{\theta_{t-1}}\Delta\theta_{t-1}^{\varepsilon z}.\end{aligned} \tag{6}$$

Since removing the participant z is equivalent to upweighting z by $-\frac{1}{n}$, we can linearly approximate the gradient change caused by removing z:

$$\Delta\mathcal{G}_t^{-z} = -\frac{1}{n}\delta_{t,z} + \alpha H_{\theta_{t-1}}(\sum_{j=1}^{t-1}\Delta\mathcal{G}_j^{-z}) \tag{7}$$

Similarly, when removing a subset of participants S, the change in gradients is $\Delta\mathcal{G}_t^{-S} = \sum_{i\in S}\Delta\mathcal{G}_t^{-i}$. □

2) For VFL: The training data is vertically partitioned and the model is distributed. Each participant i owns feature x_i of training data and a local model θ_i. The label y of training data is owned by one participant or the trusted third-party. The training dataset is $\mathcal{D} = \{(X[j], y[j]), 0 < j \leq m\}$, where $X[j] = (x_1[j], x_2[j], ..., x_n[j])^\top$. In epoch t, the global model is $\boldsymbol{\theta}_{t-1} = (\theta_{t-1,1}, \theta_{t-1,2}, ..., \theta_{t-1,n})^\top$, and participant i calculates the local result with the local parameters $\theta_{t-1,i}$ and training data, which is $\delta_{t,i} = f(\theta_{t-1,i}, x_i)$, and sends it to the trusted third-party. Aggregating local results $\Lambda_t = \{\delta_{t,1}, \delta_{t,2}, ..., \delta_{t,n}\}$, the trusted third-party computes the global loss $L(\delta_{t,1}, \delta_{t,2}, ..., \delta_{t,n})$ where L is the loss function. To facilitate analysis, we can define loss as $loss(\boldsymbol{\theta}_t)$. The global gradient is

$$\begin{aligned}\mathcal{G}_t &\overset{def}{=} \alpha_t\nabla loss(\boldsymbol{\theta}_{t-1})\\ &= \alpha_t\left(\frac{\partial loss(\boldsymbol{\theta}_{t-1})}{\partial\theta_{t-1,1}}, ..., \frac{\partial loss(\boldsymbol{\theta}_{t-1})}{\partial\theta_{t-1,n}}\right)^\top,\end{aligned}$$

where α_t is the learning rate at epoch t.

If for any x_i, $f(0, x_i) \equiv 0$, then when the model is initialized to $\mathbf{0}$, removing the participant z is equivalent to

not updating the local model of z in each epoch. That means the local output of participant z is always 0, which does not affect the calculation of loss or the training of the model.

After removing the participant z, the global model is $\boldsymbol{\theta}^{-z} \overset{def}{=} (\theta_1^{-z}, ..., \theta_z^{-z} \equiv 0, ..., \theta_n^{-z})^\top$, and the global loss is $loss(\boldsymbol{\theta}_{t-1}^{-z})$. The global gradient is

$$\begin{aligned}\mathcal{G}_t^{-z} &= \alpha_t diag(\vec{v}_z)\nabla loss(\boldsymbol{\theta}_{t-1}^{-z}) \\ &= \alpha_t diag(\vec{v}_z)\left(\frac{\partial loss(\boldsymbol{\theta}_{t-1}^{-z})}{\partial \theta_{t,1}}, ..., \frac{\partial loss(\boldsymbol{\theta}_{t-1}^{-z})}{\partial \theta_{t,n}}\right)^\top,\end{aligned}$$

where $\vec{v}_z = (v_1, ..., v_j, ..., v_n)$, if $j = z$, then $v_j = 0$, otherwise $v_j = 1$. The change in gradients is $\Delta\mathcal{G}_t^{-z} = \mathcal{G}_t^{-z} - \mathcal{G}_t$ and change in parameters is $\Delta\theta_t^{-z} = \theta_t^{-z} - \theta_z = -\sum_{j=1}^{t}\Delta\mathcal{G}_j^{-z}$.

Lemma 2. *For VFL, in epoch t, if the loss function is twice-differentiable, when participant z is removed, the change in gradients is:*

$$\Delta\mathcal{G}_t^{-z} = \mathcal{G}_t^{-z} - \mathcal{G}_t = -(E - diag(\vec{v}_z))\mathcal{G}_t - \alpha_t\Omega_t^{-z}, \quad (8)$$

where $\Omega_t^{-z} = diag(\vec{v}_z)H_{(\boldsymbol{\theta}_{t-1})}(\sum_{j=1}^{t-1}\Delta\mathcal{G}_j^{-z})$ and E is the identity matrix.

And the change satisfies additivity, namely, when a subset of participants S is removed, the change in gradients is:

$$\Delta\mathcal{G}_t^{-S} = \sum_{i\in S}\Delta\mathcal{G}_t^{-i}. \quad (9)$$

Proof Sketch. *We assume that the initialized model is $\boldsymbol{\theta}_0 = \mathbf{0}$.*

At epoch t, the change in gradients is:

$$\begin{aligned}\Delta\mathcal{G}_t^{-z} &= \mathcal{G}_t^{-z} - \mathcal{G}_t \\ &= \alpha_t diag(\vec{v}_z)\nabla loss(\boldsymbol{\theta}_{t-1}^{-z}) - \alpha_t\nabla loss(\boldsymbol{\theta}_{t-1}) \\ &\approx -(E - diag(\vec{v}_z))\mathcal{G}_t \\ &\quad - \alpha_t diag(\vec{v}_z)H_{(\boldsymbol{\theta}_{t-1})}\sum_{i=1}^{t-1}\Delta\mathcal{G}_i^{-z}.\end{aligned} \quad (10)$$

In particular, when epoch is 1,

$$\Delta\mathcal{G}_1^{-z} = -(E - diag(\vec{v}_z))\mathcal{G}_1. \quad (11)$$

Similarly, when a subset of participants S is removed, the change in gradients is $\Delta\mathcal{G}_t^{-S} = \sum_{i\in S}\Delta\mathcal{G}_t^{-i}$. □

For HFL and VFL, Lemma 1 and Lemma 2 illustrate how to measure the impact of each participant on model parameters and gradients.

D. Contribution Calculation

Based on the impact of each participant on model parameters and gradients, we can measure the impact of the participant on the loss of the validation dataset $\mathcal{D}^v$, that is the impact on the utility function.

Lemma 3. *For HFL and VFL, when the participant z is removed, the change in utility function is*

$$\Delta V^{-z} = \sum_{t=1}^{\tau}\nabla loss^v(\theta_{t-1})\Delta\mathcal{G}_t^{-z}. \quad (12)$$

And the change of utility function satisfies additivity, that is, when a set of participants S is removed, the change in utility function $\Delta V^{-S} = \sum_{i\in S}\Delta V^{-i}$.

Lemma 3 shows how to measure the impact of each participant on the utility function. Due to its additivity, we can reduce the exponential complexity of calculating the Shapley value to linear complexity and utilize the training log and the Hessian matrix of the loss to estimate the actual contribution. For HFL and VFL, the Shapley value of participant i is

$$\begin{aligned}\phi_i &= \sum_{S\subseteq\mathcal{C}}\frac{V(S) - V(S\setminus\{i\})}{n\binom{n-1}{|S|}} \\ &= -\sum_{t=1}^{\tau}\nabla loss^v(\theta_{t-1})\Delta\mathcal{G}_t^{-i}.\end{aligned} \quad (13)$$

At epoch t, the per-epoch contribution of participant i is

$$\begin{aligned}\phi_{t,i} &= -\nabla loss^v(\theta_{t-1})\Delta\mathcal{G}_t^{-i} \\ &= \nabla loss^v(\theta_{t-1})K_{t,i} + \alpha_t\nabla loss^v(\theta_{t-1})\Omega_t^{-i}\end{aligned} \quad (14)$$

where in VFL $K_{t,i} = -(E - diag(\vec{v}_i))\mathcal{G}_t$ and $\Omega_t^{-i} = diag(\vec{v}_i)H_{(\boldsymbol{\theta}_{t-1})}(\sum_{j=1}^{t-1}\Delta\mathcal{G}_j^{-i})$, and in HFL $K_{t,i} = -\frac{1}{n}\delta_{t,i}$ and $\Omega_t^{-i} = H_{\theta_{t-1}}(\sum_{j=1}^{t-1}\Delta\mathcal{G}_j{}^{-i})$.

And the contribution of participant i in the whole training process is

$$\phi_i = \sum_{t=1}^{\tau}\phi_{t,i}. \quad (15)$$

E. Complexity Analysis and Further Optimization

With the aforementioned two steps, DIG-FL enables us to highly efficiently calculate the contribution of each participant for both HFL and VFL. The per-epoch contribution of participant z at epoch t mainly consists of two terms: $\nabla loss^v(\theta_{t-1})K_{t,z}$ and $\alpha_t\nabla loss^v(\theta_{t-1})\Omega_t^{-z}$.

For calculating the first term $\nabla loss^v(\theta_{t-1})K_{t,z}$, the server needs $O(\tau np)$ operations, while participants do not need any additional computation or communication. For directly calculating the second term $\alpha_t\nabla loss^v(\theta)\Omega_t^{-z}$, the server needs to construct H_θ, the Hessian matrix of the loss function, which requires $O(\tau np^2)$ operations (p is the number of model parameters). All participants need to calculate $\mathcal{G}$ and upload H_θ and $\mathcal{G}$ to the server, which requires $O(\tau p^2)$ operations and $O(\tau p^2)$ communication cost. Considering τ and p could be large in many deep learning tasks, directly computing Eq.(14) and Eq.(15) could cause large extra computation and communication overhead. To reduce the cost, we explore the magnitude of the first and second terms and notice that the second term is much smaller than the first one due to the small coefficient α, i.e., the learning rate. After ignoring the second term, let the contribution of participant i at epoch t be:

$$\hat{\phi}_{t,i} = \nabla loss^v(\theta_{t-1})K_{t,i}. \quad (16)$$

Our experiential results in Sec.V-B (Fig.2 and Table II) show that $\hat{\phi}_{t,i}$ and $\phi_{t,i}$ are very close, and the error caused by ignoring the second term is within 5%. Therefore, in practice, we can achieve a significantly more efficient and fairly accurate approximation of the actual contribution by computing $\hat{\phi}_{t,i}$.

Optimization for HFL. Based on the above analysis, for HFL, we design two calculation methods for different application scenarios. The first method is suitable for scenarios that require high estimation accuracy, for example, the participants are several companies. The second method is suitable for scenarios where resources are severely limited, for example, the participants are personal devices or end devices.

(1)Interactive contribution evaluation. Avoid direct calculation of $H_{\theta_{t-1}}$ with second-order optimization. We use Hessian-vector products (HVP) [35], [36] to efficiently calculate Ω_t^{-z} which requires $O(\tau np)$ operations, and then compute $loss^v(\theta_{t-1})\Omega_t^{-z}$.

(2)Resource-saving contribution evaluation. In the calculation process, we take the $\hat{\phi}_{t,i}$ which ignores the second term $-\alpha_t \nabla loss^v(\theta_{t-1})\Omega_t^{-z}$ as the contribution. At this time, only the server needs to operate $O(\tau np)$, without any additional communication and calculation overhead, and the contribution of each participant can be calculated.

For VFL, each participant only holds a part of the complete model and data, and the intermediate results are transmitted to each other under encryption, which makes it impossible to calculate the Hessian matrix of model parameters. Therefore, in VFL, we also take the $\hat{\phi}_{t,i}$ as the contribution.

F. DIG-FL based Reweight Mechanism

DIG-FL is able to efficiently calculate both the per-epoch contribution and accumulative contribution of each participant during the training process for HFL and VFL. Such ability can bring a variety of new improvements to federated learning, including optimal participant selection under budget constraint, a fairer contribution-based payment mechanism, dynamic participant reweighting, etc. Here we present how to utilize DIG-FL to dynamic reweight participants to boost the convergence of FL. Participants with low contributions usually possess low-quality data (e.g., mislabeled samples and adversarial samples), which hinders the global model from achieving convergence [10]. Through the contribution from DIG-FL, we can identify the negatively influential participants and adjust the weights of participants during training which can reduce the negative impact of participants to boost model convergence.

In epoch t, the server calculates per-epoch contributions of all participants $\phi_t = \{\phi_{t,1}, \phi_{t,2}, ..., \phi_{t,n}\}$, and then rectifies $\phi_{t,i}$ to get the non-negative weight of participant i and normalizes weights of all participants:

$$\omega_{t,i} = \frac{\max(\phi_{t,i}, 0)}{\sum_i^n \max(\phi_{t,i}, 0)}. \tag{17}$$

Applying these weights to local updates from participants, the server obtains reweighted updates:

$$\tilde{\Delta}_t = \{\omega_{t,1}\delta_{t,1}, \omega_{t,2}\delta_{t,2}, ..., \omega_{t,n}\delta_{t,n}\}. \tag{18}$$

Using $\tilde{\Delta}_t$ instead of Δ_t, the server calculates the adjusted global gradient $\tilde{\mathcal{G}}_t(\mathcal{C})$ and updates the global model.

In Sec.III-C and Sec.IV-D, we theoretically analyze the convergence speed of DIG-FL based reweight mechanism for HFL and VFL.

III. DIG-FL FOR HFL

In this section, we show how to apply our approach DIG-FL in HFL without local training data being revealed to any other party, including the server. And we give two methods suitable to calculate contribution for resource-sufficient and resource-constrained situations. In addition, we show DIG-FL based reweight mechanism how to adjust weights of participants and theoretically analyze the convergence speed in HFL.

A. HFL Protocol

For HFL, in epoch t, participant i updates the current global model θ_{t-1} using local data to obtain the local model $\theta_{t-1,i}$ and sends it to the server. Let the local update of participant i be $\delta_{t,i} = \theta_{t-1} - \theta_{t-1,i}$. The server gets $\Delta_t = \{\delta_{t,1}, \delta_{t,2}, ..., \delta_{t,n}\}$. According to Lemma 1 and Eq.14, the contribution of participant i at epoch t is:

$$\begin{aligned}\phi_{t,i} &= -\nabla loss^v(\theta_{t-1})\Delta\mathcal{G}_t^{-i} \\ &= \frac{1}{n}\nabla loss^v(\theta_{t-1})\delta_{t,i} - \alpha_t \nabla loss^v(\theta_{t-1})\Omega_t^{-i},\end{aligned} \tag{19}$$

where α_t is the learning rate at epoch t and,

$$\Omega_t^{-i} = H_{\theta_{t-1}}(\sum_{j=1}^{t-1} \Delta\mathcal{G}_j{}^{-i}). \tag{20}$$

Algorithm 1: Interactive contribution evaluation.

```
The server initializes global model θ_0 and sends it to all participants.
for each epoch t ← 1, 2, .., τ do
    The server calculates the derivative v = ∇loss^v(θ_{t-1}).
    for each participant i ← 1, 2, ..., n do
        Load global model θ_{t-1};
        Calculate Ω_t^{-i} = Ĥ_{θ_{t-1}}(Σ_{j=1}^{t-1} ΔG_j^{-i}) and update θ_t with local data to get local model θ_{t,i};
        Send local model θ_{t,i} and Ĥ_{θ_{t-1}}(Σ_{j=1}^{t-1} ΔG_j^{-i}) and to the server.
        Recorde ΔG_t^{-i} = -(1/n)δ_{t,z} - α_t Ω_t^{-i}.
    The server receives all local models {θ_{t,1}, θ_{t,2}, ..., θ_{t,n}} and Ĥ_{θ_{t-1}}(Σ_{j=1}^{t-1} ΔG_j^{-i}).
    Then, the server calculates φ_{t,i} use Eq.(19) for each participant.
    The server obtains and sends the global model θ_t to all participants.
The server calculates the whole Shapley value using Eq.(15).
```

•Algorithm #1: Interactive contribution evaluation.

The cost of calculating contribution is mainly concentrated in the Hessian matrix H_θ. In this case, we design an interactive contribution evaluation method based on stochastic estimation using HVP [35]. Specifically, at epoch t, we can efficiently calculate vectors $\Omega_t^{-i} = H_{\theta_{t-1}}(\sum_{i=1}^{t-1} \Delta\mathcal{G}_i{}^{-i})$, and then compute $\nabla loss^v(\theta_{t-1})\Omega_t^{-i} = \nabla loss^v(\theta_{t-1})H_{\theta_{t-1}}(\sum_{i=1}^{t-1} \Delta\mathcal{G}_i{}^{-i})$, and finally compute $\phi_{t,i}$ for each participant. The original HVP method computes the result for every participant one by one, which incurs prohibitively heavy cost and high privacy risk [37]. We let each participant calculate locally $\hat{H}_{\theta_{t-1}}(\sum_{i=1}^{t-1} \Delta\mathcal{G}_i{}^{-i})$ and server

calculates $E[\hat{H}_{\theta_{t-1}}(\sum_{i=1}^{t-1}\Delta\mathcal{G}_i{}^{-i})]$ as an unbiased estimator of $H_{\theta_{t-1}}(\sum_{i=1}^{t-1}\Delta\mathcal{G}_i{}^{-i})$. The detail is in Algorithm 1.

•Algorithm #2: Resource-saving contribution evaluation.

In the case of limited resources, we ignore the smaller term $-\alpha\nabla loss^v(\theta_{t-1})\Omega_t^{-i}$. Then, at epoch t, the contribution of participant i is $\phi_{t,i} \approx \frac{1}{n}\nabla loss^v(\theta_{t-1})\delta_{t,z}$. At this time, only the server needs to perform the operation of $O(np)$, without any additional communication and calculation overhead, and the contribution of each participant can be calculated. The procedure is detailed in Algorithm 2.

Algorithm 2: Resource-saving contribution evaluation.

```
The server initializes global model θ_0 and sends it to participants.
for each epoch t ← 1, 2, .., τ do
    The server calculates the derivative v = ∇loss^v(θ_{t−1}).
    for each participant i ← 1, 2, ..., n do
        Load global model θ_{t−1};
        Update θ_t with local data to get local model θ_{t−1,i};
        Send local model θ_{t−1,i} to the server.
    The server receives all local models {θ_{t−1,1}, θ_{t−1,2}, ..., θ_{t−1,n}};
    The server calculates the φ_{t,i} ≈ (1/n)∇loss^v(θ_{t−1})δ_{t,z} for each participant;
    The server obtains and sends the global model θ_t to all participants.
The server calculates the whole Shapley value using Eq.(15).
```

B. Privacy Analysis

For Algorithm #1, using local data, each participant i computes local model $\theta_{t,i}$ and $\hat{H}_{\theta_{t-1}}(\sum_{j=1}^{t-1}\Delta\mathcal{G}_j{}^{-i})$ and sends them to the server. The server performs aggregation and sends the global model to participants. No local training data is transmitted/exposed to any other parties. Hence, Algorithm #1 meets the level-1 privacy definition in Sec. II-A. Especially, compared with conventional HFL, which uploads only local model, Algorithm #1 requires each participant to additionally send the mean value of the Hessian matrix to the server. There are some recent works using gradients to recover the original data [38], [39], but no work has successfully recovered the original data using Hessian matrix. What can be inferred from the mean value of the Hessian matrix is still an open problem, therefore transmitting it may introduce an additional privacy risk in the future. Algorithm #2 uses training logs to conduct contribution measurement. As conventional HFL, participants only uploads local models. No local training data is uploaded or accessed by other parties, and no extra transmission is required. Hence, Algorithm #2 meets the level-2 privacy definition in Sec. II-A.

C. DIG-FL based Reweight Mechanism for HFL

The server reweights global gradient according to per-epoch contribution:

$$\tilde{\mathcal{G}}_t(\mathcal{C}) = \sum_{i=1}^{n}\omega_{t,i}\delta_{t,i}. \tag{21}$$

The server obtains the global model:

$$\theta_t = \theta_{t-1} - \tilde{\mathcal{G}}_t(\mathcal{C}) = \theta_t - \sum_{i=1}^{n}\omega_{t,i}\delta_{t,i}. \tag{22}$$

Let the server possess a validation dataset, we prove that the HFL algorithm with our DIG-FL based reweight mechanism can make the loss function on the validation dataset monotonically decrease and analyze the convergence rate, which is $O(log\frac{1}{\epsilon^2})$.

Lemma 4. *Suppose the loss function on the validation dataset $loss^v(\theta)$ is Lipschitz-smooth with constant L, and the modulus of local update $||\delta_{t,i}||$ has an upper bound δ. Let the learning rate α_t satisfy $\alpha_t \le \frac{2}{L\delta^2}$. The validation loss always monotonically decreases, i.e.,*

$$loss^v(\theta_{t+1}) \le loss^v(\theta_t). \tag{23}$$

And,

$$\min_{1\le t\le\tau}||\nabla loss^v(\theta_t)|| \le \frac{\xi}{\sqrt{\tau}}, \tag{24}$$

where ξ is a constant independent of the convergence process.

IV. DIG-FL for VFL

Here, we first show how to apply DIG-FL to VFL by designing a VFL protocol. Then we give a privacy-preserving participant contribution calculation method based on encryption methods. Finally, we present DIG-FL based reweight mechanism for dynamically reweighting participants in training and theoretically validating its higher convergence speed.

A. VFL Protocol

For VFL, there are usually a trusted third-party and n participants $\mathcal{C} = \{1, 2, ..., n\}$. The trusted third-party generates and distributes encryption key pairs. The training data is vertically partitioned and thus the model is distributed. Participant i owns feature x_i of training data and a local model θ_i. The label y of training data is owned by one participant or the trusted third-party. The training dataset is $\mathcal{D} = \{(X[i], y[i]), 0 < i \le m\}$, where $X[i] = (x_1[i], x_2[i], ..., x_n[i])^\top$ and the global model is $\boldsymbol{\theta} = (\theta_1, \theta_2, ..., \theta_n)^\top$. Model training starts from epoch 1, and in epoch t, participant i calculates the local result with the local parameters and training data, which is $\delta_{t,i} = f(\theta_{t-1,i}, x_i)$, and sends it to the trusted third-party.

To protect the data privacy of participants, the training is usually carried out on ciphertexts. Our design can easily apply to various VFL systems. Here we first ignore the details of encryption and present the general method to compute contributions and weights, and then give a concrete example in Sec.IV-B.

At epoch t, the global gradient:

$$\mathcal{G}_t = \alpha_t\left(\frac{\partial loss(\boldsymbol{\theta}_{t-1})}{\partial\theta_{t-1,1}}, ..., \frac{\partial loss(\boldsymbol{\theta}_{t-1})}{\partial\theta_{t-1,n}}\right)^\top. \tag{25}$$

The per-epoch contribution of participant i is:

$$\begin{aligned}\phi_{t,i} &= -\nabla loss^v(\theta_{t-1})\Delta\mathcal{G}_t^{-i}\\ &= \nabla loss^v(\theta_{t-1})((E - diag(\vec{v}_i))\mathcal{G}_t + \alpha_t\Omega_t^{-i}),\end{aligned} \tag{26}$$

where $\Omega_t^{-i} = diag(\vec{v}_z)H_{(\boldsymbol{\theta}_{t-1})}(\sum_{j=1}^{t-1}\Delta\mathcal{G}_j^{-i})$ and $\vec{v}_i = (v_1, ..., v_j, ..., v_n)$, if $j = i$, $v_j = 0$, else $v_j = 1$.

And as aforementioned in Sec.II-E, we ignore the second term when calculating contribution. The contribution of participant i at epoch t is:

$$\phi_{t,i} = \nabla loss^v(\boldsymbol{\theta}_{t-1})(E - diag(\vec{v}_z))\mathcal{G}_t. \tag{27}$$

B. Running Example Protocol

There are different VFL frameworks in industry and academia, e.g., [3], [6], [8]. Here we adapt the well-known vertical linear regression proposed by [3] a running example to show how to adapt our approach in VFL.

For vertical linear regression, participant 1 owns local model θ_1, local data x_1 and label y; participant 2 owns local model θ_2 and local x_2; and the trusted third-party generates key pairs. They jointly train a linear regression model $\boldsymbol{\theta} = (\theta_1, \theta_2)^\top$. The training data is $\mathcal{D} = \{(X[i], y[i]), 0 < i \leq m\}$, and the validation data is $\mathcal{D}^v = \{(X[i], y[i]), 0 < i \leq m^v\}$, where $X[i] = (x_1[i], x_2[2])^\top$.

To apply DIG-FL to this training process, firstly, participants calculate $\nabla loss(\boldsymbol{\theta}) = (\frac{\partial loss}{\partial \theta_1}, \frac{\partial loss}{\partial \theta_2})$ under the privacy framework of [3], which uses additive homomorphic encryption, denoted as $[[\cdot]]$. Specifically, we have

$$loss(\boldsymbol{\theta}) = \sum_{\mathcal{D}} (\theta_1 x_1 + \theta_2 x_2 - y)^2, \tag{28}$$

where $loss$ is the loss function on the training data, and

$$\frac{\partial loss}{\partial \theta_i} = 2 \sum_{\mathcal{D}} (\theta_1 x_1 + \theta_2 x_2 - y) x_i. \tag{29}$$

Letting $u_1 = \theta_1 x_1$, $u_2 = \theta_2 x_2$, the encrypted gradient is

$$[[\frac{\partial loss}{\partial \theta_i}]] = [[2 \sum_{\mathcal{D}} (u_1 + u_2 - y) x_i]]. \tag{30}$$

Let $[[d]] = [[u_1 - y]] + [[u_2]]$, then the process of calculating $\frac{\partial loss}{\partial \theta_1}$ and $\frac{\partial loss}{\partial \theta_2}$ usually includes the following five steps:

1) The trusted third-party creates a key pair and sends the public key to participant 1 and 2.
2) Participant 1 computes $[[u_1 - y]]$ and sends it to participant 2.
3) Participant 2 computes $[[u_2]]$ and $[[d]]$, and sends $[[d]]$ to participant 1.
4) Participant 1 initializes M_1, computes $[[\frac{\partial loss}{\partial \theta_1}]]$ and sends $[[\frac{\partial loss}{\partial \theta_1} + M_1]]$ to the trusted third-party; participant 2 initializes M_2, computes $[[\frac{\partial loss}{\partial \theta_2}]]$ and sends $[[\frac{\partial loss}{\partial \theta_2} + M_2]]$ to the trusted third-party.
5) The trusted third-party decrypts, sends $\frac{\partial loss}{\partial \theta_1} + M_1$ to participant 1, and $\frac{\partial loss}{\partial \theta_2} + M_2$ to to participant 2.

To prevent the trusted third-party to learn information from participant 1 or participant 1 in this process, participants can hide their intermediate results (gradients) by adding encrypted random masks M_1 and M_2.

And $\mathcal{G}(\{1\}) \overset{def}{=} (E - diag(\vec{v}_1))\nabla loss(\boldsymbol{\theta}) = (\frac{\partial loss}{\partial \theta_1}, 0)^\top$ and $\mathcal{G}(\{2\}) \overset{def}{=} (E - diag(\vec{v}_2))\nabla loss(\boldsymbol{\theta}) = (0, \frac{\partial loss}{\partial \theta_2})^\top$.

Secondly, following the same steps of calculating $\nabla loss(\boldsymbol{\theta})$, participants calculate the validation gradient $\nabla loss^v(\boldsymbol{\theta})$ under the privacy framework of [3].

The complete training process is shown in Algorithm 3.

Algorithm 3: DIG-FL for Vertical Linear Regression

```
Participant 1 initializes local model θ_{0,1} and participant 2
  initializes local model θ_{0,2}. The trusted third-party P creates
  an encryption key pair and sends the public key to participants.
for each round t ← 1, 2, .., τ do
    Jointly calculate G_t({1}) and G_t({2}).
    Jointly calculate the validation gradient ∇loss^v(θ_t).
    Jointly calculates the shapley values φ_t = {φ_{t,1}, φ_{t,2}}
      using Eq.(27) ;
    Participants update local models.
The third-party calculates the Shapley value using Eq.(15).
```

We can also apply DIG-FL to various VFL frameworks including [6], [8] and we will present evaluations in Sec.V.

C. Privacy Analysis

Algorithm #3 uses the vertical linear regression proposed by [3] as an example to show how to apply DIG-FL to VFL. According to the security analysis in [3], during the training process itself, any party can learn nothing from other parties beyond what is revealed by his/her own input and output. In addition to training model, Algorithm #3 needs participants to cooperatally compute $\mathcal{G}_t(\{1\})$, $\mathcal{G}_t(\{2\})$, and $\nabla loss^v(\theta_t)$ to estimate shapley values. During the calculation, since participants add encrypted random masks to hind their gradients, the trusted third-party can not infer or learn any information from participants [40]. Secondly, since the intermediate results transmitted are all encrypted by Paillier with key length 1024, participant 1 can only get its own gradients. That is not enough for participant 1 to learn or infer any information from participant 2, due to the inability of solving n equations in more than n unknowns[41]. Similarly, participant 2 can not learn any information from participant 1. At the end of Algorithm #3, participant 1 or participant 2 obtain the model parameters associated only with its own features. Therefore, Algorithm #3 meet the privacy definition in Sec.II-A.

D. DIG-FL based Reweight Mechanism for VFL

The trusted third-party reweights participants and gets the tuned global gradient:

$$\tilde{\mathcal{G}}_t = \left(\omega_{t,1}\frac{\partial loss}{\partial \theta_{t,1}}, \omega_{t,2}\frac{\partial loss}{\partial \theta_{t,2}}, ..., \omega_{t,n}\frac{\partial loss}{\partial \theta_{t,n}}\right)^\top. \tag{31}$$

And the updated model is $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha\tilde{\mathcal{G}}_t$.

Let the there is a validation dataset, under some conditions, we prove that VFL algorithms with DIG-FL based reweight mechanism can make the loss function of the validation dataset converge to the critical point and we also theoretically prove its sub-liner convergence rate.

Lemma 5. *Suppose the validation loss function $loss^v(\theta)$ is Lipschitz-smooth with constant L, and the gradient of training data has an upper bound δ. Let the learning rate α_t satisfies $\alpha_t \leq \frac{2}{Ln\delta^2}$, where n is the number of participant. Then, the validation loss always monotonically decreases, i.e.,*

$$loss^v(\theta_{t+1}) \leq loss^v(\theta_t). \tag{32}$$

And,

$$\min_{1\le t\le \tau} ||\nabla loss^v(\theta_t)|| \le \frac{\xi}{\sqrt{\tau}}, \tag{33}$$

where ξ is a constant independent of the convergence process.

V. Evaluations

In this section, by experiments, we demonstrate the rationality of omitting the second term in Sec.II-E. Then we measure the effectiveness of DIG-FL approximating the actual Shapley value. In addition, we verify the superiority of our work by comparing with existing methods. Finally, we show that DIG-FL based reweight mechanism really can improve the global model with higher accuracy and faster convergence speed.

A. Experimental Configuration

1) Datasets: For HFL, we use two public image datasets MNIST and CIFAR10, and two crawled datasets to test our methods in real-world scenarios. The crawled datasets include: a) MOTOR: it consists of 11,000 images from two classes: motorcycle and non-motorcycle; b) REAL: there are 110,000 images crawled by using 10 keywords including Banana, Bowl, Bread, Crab, Elephant, Frog, House, Pig, Rabbit, and Snail. For VFL, we used ten public tabular datasets. Details of these datasets are presented in Table I. For each dataset, we first randomly extracted 10% of the training data as the validation dataset, and distributed the remaining training data to the participants.

TABLE I
Datasets.

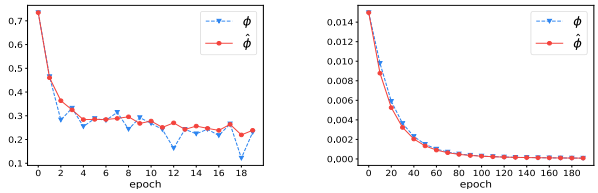
Task	Dataset	Size	Description
HFL	$\mathcal{D}_M$	70,000	MNIST [42]
	$\mathcal{D}_C$	60,000	CIFAR10 [43]
	$\mathcal{D}_O$	11,000	MOTOR [16]
	$\mathcal{D}_R$	110,000	REAL [16]
VFL	$\mathcal{D}_B$	506*14	Boston house-prices [29]
	$\mathcal{D}_D$	442*11	Diabetes [29]
	$\mathcal{D}_{Wq}$	4898*12	Wine quality [29]
	$\mathcal{D}_S$	17379*15	Seoul bike sharing Demand [29]
	$\mathcal{D}_{Ca}$	20641*9	California house-prices [44]
	$\mathcal{D}_I$	150*5	Iris dataset [29]
	$\mathcal{D}_W$	173*14	Wine dataset [29]
	$\mathcal{D}_{Bc}$	569*31	Breast cancer dataset [29]
	$\mathcal{D}_{Cc}$	30000*23	Default of credit card clients [29]
	$\mathcal{D}_A$	48842*15	Adult dataset [29]

2) Deep learning models: For HFL, we implemented the typical federated optimization algorithm `FedSGD` [4] and four popular deep learning models, namely `HFL-CNN-MNIST`, `HFL-CNN-CIFAR`, `HFL-CNN-MOTOR` and `HFL-CNN-REAL` for image classification. For VFL, we implemented two models, namely a vertical linear regression model `VFL-LinReg` [3] and a vertical logistical regression model `VFL-LogReg` [3].

3) Metrics: We test our framework DIG-FL from two aspects: accuracy and cost. The accuracy is quantified by Pearson's Correlation Coefficient (PCC) between the estimated Shapley value of DIG-FL and actual Shapley value. The cost includes computation cost and communication cost. Computation cost is quantified by the time(s) that algorithm runs and communication cost is quantified by the amount of data(MB) that the server interacts with the participants.

B. Error of Ignoring the Second Term

In Sec.II-E, we propose to ignore the second term $\alpha\nabla loss^v(\theta)\Omega$ when calculating the contribution to save resources. Then we test the error of ignoring the second term on 14 datasets for HFL and VFL.



(a) per-epoch contribution for HFL. (b) per-epoch contribution for VFL.

Fig. 2. ϕ is the whole contirbution, while $\hat{\phi}$ is the contribution ignoring the second term.

TABLE II
The error of ignoring the second term.

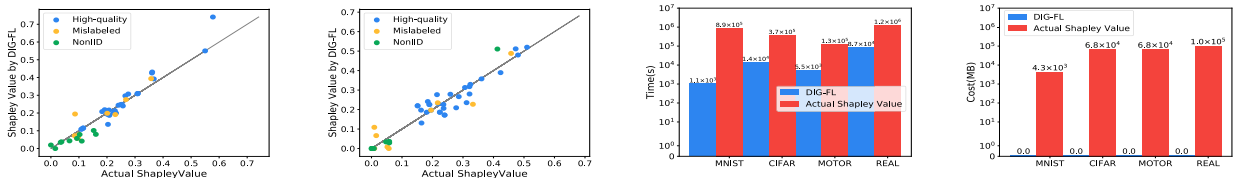
Model	Dataset	ϕ	$\hat{\phi}$	$\lvert\frac{\phi-\hat{\phi}}{\phi}\rvert$
HFL-CNN-MNIST	$\mathcal{D}_M$	2.771	2.786	0.54%
HFL-CNN-CIFAR	$\mathcal{D}_C$	7.094	7.306	2.98%
HFL-CNN-MOTOR	$\mathcal{D}_O$	5.81	6.09	4.82%
HFL-CNN-REAL	$\mathcal{D}_R$	3.709	3.888	4.82%
VFL-LinReg	$\mathcal{D}_B$	0.367	0.376	2.45%
	$\mathcal{D}_D$	0.250	0.255	2.00%
	$\mathcal{D}_{Wq}$	0.195	0.196	0.51%
	$\mathcal{D}_S$	0.459	0.477	3.92%
	$\mathcal{D}_{Ca}$	0.262	0.261	0.38%
VFL-LogReg	$\mathcal{D}_I$	0.343	0.347	1.16%
	$\mathcal{D}_W$	0.104	0.109	4.81%
	$\mathcal{D}_{Bc}$	0.0649	0.0665	2.47%
	$\mathcal{D}_{Cc}$	0.0498	0.0475	4.62%
	$\mathcal{D}_A$	0.194	0.195	0.52%

Fig. 2 and Table II show that, for HFL and VFL, the error caused by ignoring the second term is within 5%, which is acceptable.

C. DIG-FL v.s. Actual Shapley Value

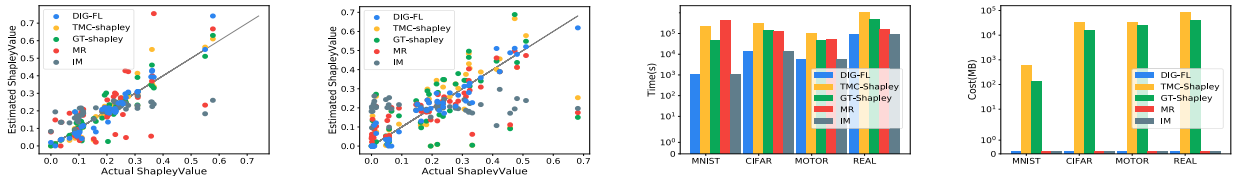
We compare the Shapley value estimated by DIG-FL with the actual Shapley value. The actual Shapley value is computed by performing 2^n retraining (n is the number of participants) and using Eq.(2) as the utility function.

1) For HFL: On the four public datasets MNIST, CIFAR10, MOTOR, REAL, we generated two typical types of low-contribution participants: 1) participant holds mislabeled data; 2) participant holds nonIID data. We set m as the number of low-contribution participants. With mislabeled data, we first evenly distributed $\mathcal{D}_M$ ($\mathcal{D}_C$, $\mathcal{D}_O$ and $\mathcal{D}_R$) to n participants, and then for m out of n participants, we replaced the labels of 50% (or 30%) of their training samples with random incorrect labels from the same dataset. With nonIID data, we divided images into shards of different categories. We evenly assigned



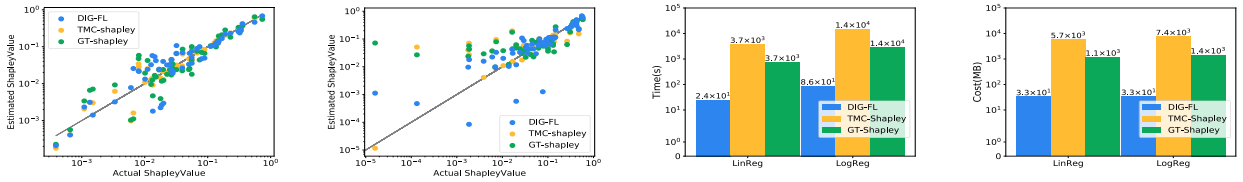
(a) HFL-CNN-CIFAR (b) HFL-CNN-MOTOR (c) Computation cost (d) Communication cost

Fig. 3. Accuracy and cost of estimating Shapley values by DIG-FL and computing actual Shapley value for HFL.



(a) HFL-CNN-CIFAR (b) HFL-CNN-MOTOR (c) Computation cost (d) Communication cost

Fig. 4. DIG-FL v.s. TMC-shapley, GT-shapley, MR and IM in HFL.



(a) VFL-LinReg (b) VFL-LogReg (c) Computation cost (d) Communication cost

Fig. 5. DIG-FL v.s. TMC-shapley and GT-shapley in VFL.

shards from all categories (i.e., IID data) to $n-m$ participants, and for the rest m participants we randomly assigned them incomplete categories of shards (i.e., non-IID data with only 1 to 9 categories out of 10 categories). For $\mathcal{D}_M$, $n = 10$ and m varied from 0 to 9; for $\mathcal{D}_C$, $\mathcal{D}_O$ and $\mathcal{D}_R$, $n = 5$ and m varied from 0 to 4.

Fig. 3 (a) and (b) depict the Shapley values estimated by DIG-FL and the corresponding actual values of all participants in all cases (with different m). The results show that the estimated Shapley values are very close to the actual values, with 0.968 Pearson's Correlation Coefficient (PCC) on MNIST, 0.935 PCC on CIFAR10, 0.952 PCC on MOTOR and 0.833 PCC on REAL. Participants with high-quality (error-free and IID) data have obviously greater Shapley values than participants with mislabeled or nonIID data.

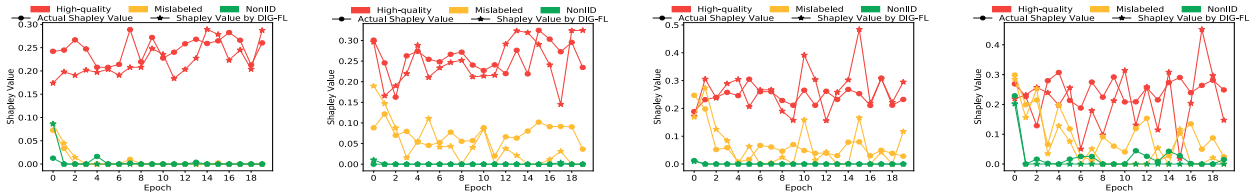
More importantly, Fig. 3 (c) and (d) show that DIG-FL dramatically saves computation cost by orders of magnitude, e.g., from 8.9×10^5 to 1.1×10^3 on MNIST, and does not cause any communication overhead. Therefore, DIG-FL can accurately and highly efficiently estimate the Shapley value.

2) For VFL: The detailed experiment settings and results are presented in Table III. The results show that the Shapley values estimated by DIG-FL and the actual Shapley values are very close, achieving 0.987 average PCC for `VFL-LinReg` and 0.940 average PCC for `VFL-LogReg`. Moreover, compared with computing the actual Shapley values, DIG-FL dramatically reduces the time cost by orders of magnitude in all cases. For example, DIG-FL reduced the time cost from 76,584.7 to 13.77 seconds on $\mathcal{D}_S$, while the PCC is 0.998.

TABLE III
Settings and performance of DIG-FL for VFL. n is the number of participants, and PCC is between the estimated Shapley value and the actual value. $T_{DIG-FL}(s)$ and $T_{Actual}(s)$ are time cost of DIG-FL and calculating the actual Shapley value, respectively.

Model	**Dataset**	n	PCC	T_{DIG-FL}	T_{Actual}
`VFL-LinReg`	$\mathcal{D}_B$	13	0.978	1.09	3802.73
	$\mathcal{D}_D$	10	0.986	1.032	462.66
	$\mathcal{D}_{Wq}$	11	0.987	1.68	1279.1
	$\mathcal{D}_S$	14	0.998	13.77	76584.7
	$\mathcal{D}_{Ca}$	8	0.984	7.02	643.05
`VFL-LogReg`	$\mathcal{D}_I$	4	0.981	0.132	1.23
	$\mathcal{D}_W$	13	0.941	1.482	3485.81
	$\mathcal{D}_{Bc}$	15	0.954	3.93	21120.25
	$\mathcal{D}_{Cc}$	11	0.921	60.73	44406.34
	$\mathcal{D}_A$	14	0.901	20.07	120028.81

3) DIG-FL v.s. Actual Shapley Value for each epoch: The above subsections compare the estimated and actual Shapley values for the entire training process. Here, we further investigate them for each epoch. For HFL, in each epoch, we use model performance improvement caused by the gradient sent by each participant to the server as the utility function of the actual Shapley value. A participant leaving the FL system is equivalent to ignoring his/her uploaded gradient when the server performs aggregation. As mentioned in Sec. V-C1, we generated three typical types of participants: 1) participant holding high-quality data; 2) participant holding mislabeled data; 3) participant holding non-IID data, on four datasets MNIST, CIFAR10, MOTOR, REAL. For all datasets,



(a) MNIST (b) CIFAR (c) MOTOR (d) REAL

Fig. 6. DIG-FL v.s. Actual Shapley Value for each epoch. Lines with dots are for the actual Shapley values. Lines with stars are for the estimated Shapley values. The colors indicate types of participants. Red, yellow and green mean participants holding high-quality, mislabelled and non-IID data, respectively.

we set the number of participants to 5, one of which holds non-IID data and one holds mislabeled data. During the training, once some participants leave, the rest participants and the server continuously train the model. For VFL, each participant owns some features and a part of the complete model. Once a participant leaves, the structure of the model will change, therefore the rest participants cannot continue the model training or inference. So we only present the Shapley values for each epoch in the HFL scenario. Fig. 6 depicts that, for all epochs, our estimated Shaley values are very close to the actual shaley values for three types of participants on four datasets. It also shows that participants with high-quality (error-free and IID) data have greater Shapley values than that of participants with mislabeled data, participants with non-IID data have the smallest Shapley values. In each epoch, the communication cost for calculating the actual and estimated Shapley values are the same, but the computation cost for the actual value is 2^n times higher than that for our estimated value, which is similar to Fig. 3 (c).

D. Comparison with Benchmark Methods

For centralized learning, there exists two state-of-the-art methods for calculating the Shapley value: (1) Truncated Monte Carlo Shapley (TMC-shapley) [20]; (2) GT-shapley [21]. which proposes a repertoire of efficient algorithms for approximating the Shapley value. We apply these methods to HFL and VFL. Adopting the same settings in their work, we set the rounds of retraining for TMC-shapley to $n^2 \log n$ and GT-shapley to $n(\log n)^2$, where n is the number of participants. For HFL, Song et al. [23] propose the method Multi-Rounds Reconstruction based Algorithm (MR). Zhang et al. [18] use the local model update projected onto the finally global model to measure contribution(IM). For VFL, to the best of our knowledge, no method can be applied to various horizontal federated learning frameworks to measure contributions. Therefore, we compare DIG-FL with four methods for calculating the Shapley value in HFL and two methods in VFL. The datasets and models are the same as in Sec. V-C.

1) Comparison with benchmark methods in HFL: we compare DIG-FL with four methods: TMC-shapley, GT-shapley, MR and IM for calculating the Shapley value.

Fig. 4 (a) and (b), and Table IV show that DIG-FL achieves better estimation accuracy, and the average PCC between estimated Shapley values and the actual Shapley values is 0.922 for DIG-FL, 0.860 for TMC-shapley, 0.826 for GT-shapley and 0.832 for MR. Fig. 4 (c) and (d) show that DIG-FL dramatically saves computation cost by orders of magnitude, and does not cause any communication overhead.

TABLE IV
PCC between Shapley values estimate by different methods and the actual Shapley values in HFL.

Datasets	DIG-FL	TMC-shapley	GT-shapley	MR	IM
MNIST	0.968	0.917	0.865	0.912	0.681
CIFAR	0.935	0.903	0.874	0.776	0.673
MOTOR	0.852	0.816	0.753	0.778	0.319
REAL	0.833	0.802	0.822	0.857	0.213

2) Comparison with benchmark methods in VFL: we compare DIG-FL with two methods: TMC-shapley, GT-shapley for calculating the Shapley value.

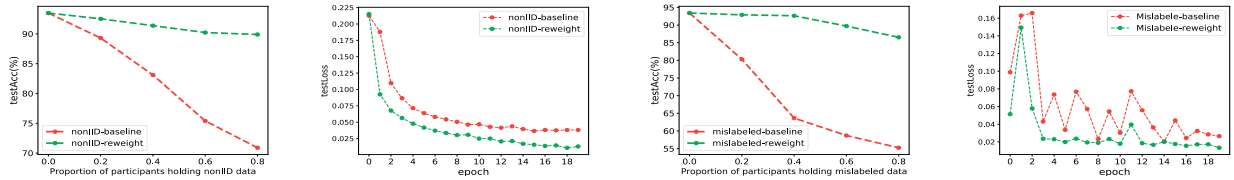
TABLE V
PCC between Shapley values estimate by different methods and the actual Shapley values in VFL.

Model	Dataset	DIG-FL	TMC-shapley	GT-shapley
VFL-LinReg	$\mathcal{D}_B$	0.978	0.994	0.912
	$\mathcal{D}_D$	0.986	0.987	0.936
	$\mathcal{D}_{Wq}$	0.987	0.994	0.971
	$\mathcal{D}_S$	0.994	0.996	0.974
	$\mathcal{D}_{Ca}$	0.983	0.995	0.983
VFL-LogReg	$\mathcal{D}_I$	0.981	0.987	0.986
	$\mathcal{D}_W$	0.941	0.873	0.862
	$\mathcal{D}_{Bc}$	0.954	0.892	0.853
	$\mathcal{D}_{Cc}$	0.921	0.958	0.863
	$\mathcal{D}_A$	0.901	0.896	0.829

Fig. 5 (a) and (b) show that DIG-FL achieves better estimation accuracy. Table V shows PCC between estimated Shapley values and the actual Shapley value on 10 datasets , and the average is 0.963 for DIG-FL, 0.957 for TMC-shapley and 0.917 for GT-shapley. Fig. 5 (c) and (d) illustrate that DIG-FL significantly outperforms two existing methods in both computation and communication costs, reducing cost by orders of magnitude.

E. Effect of Reweight Mechanism

When all participants hold high-quality data, the global model converges well. In this case, it is unnecessary to reweight participants. To evaluate the effect of DIG-FL based reweight mechanism, as aforementioned in Sec. V-C, we consider two settings: 1)Non-IID setting: we use two datasets MNIST and CIFAR10 and let some participants hold non-IID data. 2)Mislabeled setting: we use two datasets MOTOR and REAL and let some participants hold mislabeled data. We use



(a) TestACC of CIFAR (b) Loss of CIFAR (c) TestAcc of MOTOR (d) Loss of MOTOR

Fig. 7. The effect of reweight mechanism on model convergence.

the classic federated optimization algorithm `FedSGD` [4] as a baseline.

Due to space limitations, here we only present the results of two datasets CIFAR10 and MOTOR, which are similar to the results on MNIST and REAL, respectively. Fig. 7 (a) and (c) show that, without reweight, the performance of the global model severely deteriorates as the proportion of participants holding non-IID or mislabeled data increases. DIG-FL based reweight mechanism can significantly mitigate the negative impact of low-quality participants, so as to obviously improve the test accuracy and boosts the model convergence. For example, on CIFAR10, when four out of five participants have non-IID data, our reweight mechanism raises the test accuracy from 70.9% to 89.9%. on MOTOR, when four out of five participants have mislabeled data, the reweight mechanism raises the test accuracy from 55.2% to 86.5%. As shown in Figs. 7 (b) and (d), the reweight mechanism also obliviously boosts and stabilizes the model convergence.

VI. Related Work and Preliminaries

A. Impact based on Influence Function

Influence function is a classic technique from robust statistics [13], which tells us how the model parameters change by upweighting a training point by an infinitesimal amount. In centralized deep learning, a collection of works use influence functions to interpret model behaviors and trace predictions back to the training data [12], [13]. Some methods proceed to assign importance weights to training samples so as to improve the performance of deep learning models [45], [14], [46]. Those methods are designed for centralized learning and cannot be adopted by FL due to two main reasons: first, they require access to the training data, while the local data in FL is invisible; second, influence functions require expensive second derivative calculations for each data point, which is unaffordable for many FL systems.

Recently, Xue et al. [15] and Li et al. [16] use influence function to understand the influence of individual participants in HFL, which still needs participants to calculate and upload their Hessian matrices or part of Hessian matrices. Applying these methods to calculate contribution requires exponentially calculation of influence function, thus resulting in large extra computation and communication overhead. Moreover, all those influence function based methods are only applicable to HFL, not to VFL.

B. Contribution based on Shapley Value

Shapley Value [19] distributes cooperation benefits fairly by considering the contributions of each participant. It defines a unique distribution among the participants of the total surplus generated by the coalition and has many appealing properties, such as fairness, rationality, symmetry, and linearity.

In the machine learning area, the most recent work focus on reducing the cost of computing the Shapley value. Ghorbani et al.[20] propose a Shapley value based approach to quantify the contributions of training data points to a deep learning model, and design two methods to reduce computation cost, namely Truncated Monte Carlo Shapley and Gradient Shapley. Jia et al.[21] develop a repertoire of sample-based techniques for estimating the Shapley values of data points. Though those methods make efforts to improve the efficiency, they still require repeated leave-one-out model retraining to calculate the Shapley value and thus will impose prohibitively expensive computation and communication cost for FL systems.

There are few works [24], [23] using the Shapley value to measure contributions of participants for FL. For HFL, Song et al.[23] propose two methods One-Rounds Reconstruction based Algorithm(OR) and Multi-Rounds Reconstruction based Algorithm (MR). The second method calculates contributions in each training round and then aggregates them to get the final result without retraining the model. But it needs to exponentially test model performance, and thus imposes expensive computation cost. For VFL, Wang et al.[24] propose a method to measure participants' contribution. However, it needs to access and permutate the features of all participants, which introduces extra severe privacy risk.

VII. Conclusion

In this work, we propose an approach DIG-FL to measure the contribution of each participant, by estimating his/her Shapley value, for both HFL and VFL with minimal extra cost. We theoretically show that DIG-FL can accurately approximate the actual Shapley value using only training logs. In addition, we propose a DIG-FL based reweight mechanism to train more robust FL models when there is a large portion of participants possessing non-IID or mislabeled data. Our approach can be adopted to locate adversarial training data, optimal participant selection under budget constraint, or design fairer incentive/payment mechanisms for FL applications.

VIII. Acknowledgment

Lan Zhang is the corresponding author. This research is supported by the National Key R&D Program of China 2021YFB2900103, China National Natural Science Foundation with No. 61822209, No. 61932016, No. 62132018. This work was partially supported by Tencent Marketing Solution Rhino-Bird Focused Research Program.

REFERENCES

[1] J. Han and Y. Liu, "Rumor riding: Anonymizing unstructured peer-to-peer systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, pp. 464–475, 2011.

[2] L. Zhang, T. Jung, K. Liu, X. Li, X. Ding, J. Gu, and Y. Liu, "Pic: Enable large-scale privacy preserving content-based image search on cloud," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, pp. 3258–3271, 2017.

[3] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *TIST*, 2019.

[4] B. McMahan, E. Moore, and Ramage, "Communication-efficient learning of deep networks from decentralized data," in *ICML*, 2017.

[5] K. Muhammad, Q. Wang, D. O'Reilly-Morgan, E. Tragos, B. Smyth, N. Hurley, J. Geraci, and A. Lawlor, "Fedfast: Going beyond average for faster training of federated recommender systems," in *SIGKDD*, 2020.

[6] Y. Hu, D. Niu, J. Yang, and S. Zhou, "Fdml: A collaborative machine learning framework for distributed features," in *SIGKDD*, 2019.

[7] B. Gu, Z. Dang, X. Li, and H. Huang, "Federated doubly stochastic kernel learning for vertically partitioned data," in *SIGKDD*, 2020.

[8] Y. Liu, Y. Kang, X. wei Zhang, L. Li, Y. Cheng, T. Chen, M. Hong, and Q. Yang, "A communication efficient collaborative learning framework for distributed features," *arXiv:Learning*, 2019.

[9] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[10] A. Shafahi and W. R. Huang, "Poison frogs! targeted clean-label poisoning attacks on neural networks," in *NIPS*, 2018.

[11] S. Mehnaz and E. Bertino, "Privacy-preserving real-time anomaly detection using edge computing," in *IEEE ICDE*, 2020, pp. 469–480.

[12] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," in *ICML*, 2017.

[13] R. D. Cook and S. Weisberg, *Residuals and influence in regression*. New York: Chapman and Hall, 1982.

[14] M. Ren, W. Zeng, B. Yang, and R. Urtasun, "Learning to reweight examples for robust deep learning," in *ICML*, 2018.

[15] Y. Xue, C. Niu, Z. Zheng, S. Tang, C. Lv, F. Wu, and G. Chen, "Toward understanding the influence of individual clients in federated learning," in *AAAI*, 2021.

[16] A. Li, L. Zhang, J. Wang, J. Tan, F. Han, Y. Qin, N. Freris, and X.-Y. Li, "Efficient federated-learning model debugging," *ICDE*, 2021.

[17] J. W. F. H. X.-Y. L. Anran Li, Lan Zhang, "Privacy-preserving efficient federated-learning model debugging," *IEEE Transactions on Parallel and Distributed Systems*, 2021.

[18] J. Zhang, Y. Wu, and R. Pan, "Incentive mechanism for horizontal federated learning based on reputation and reverse auction," *Proceedings of the Web Conference 2021*, 2021.

[19] L. S. Shapley, "A value for n-person games," 1988.

[20] A. Ghorbani and J. Zou, "Data shapley: Equitable valuation of data for machine learning," in *ICML*, 2019.

[21] R. Jia, D. Dao, B. Wang, F. A. Hubis, N. Hynes, N. M. Gürel, B. Li, C. Zhang, D. Song, and C. Spanos, "Towards efficient data valuation based on the shapley value," in *AISTATS*, 2019.

[22] S. Wang, "When edge meets learning: Adaptive control for resource-constrained distributed machine learning," in *INFOCOM*, 2018.

[23] T. Song, Y. Tong, and S. Wei, "Profit allocation for federated learning," in *Big Data*, 2019.

[24] G. Wang, C. X. Dang, and Z. Zhou, "Measure contribution of participants in federated learning," in *Big Data*, 2019.

[25] L. Zhang, X. Li, K. Liu, C. Liu, X. Ding, and Y. Liu, "Cloak of invisibility: Privacy-friendly photo capturing and sharing system," *IEEE Transactions on Mobile Computing*, vol. 18, pp. 2488–2501, 2019.

[26] L. Zhang, X. Li, K. Liu, T. Jung, and Y. Liu, "Message in a sealed bottle: Privacy preserving friending in mobile social networks," *IEEE Transactions on Mobile Computing*, vol. 14, pp. 1888–1902, 2015.

[27] L. Zhang, X. Li, Y. Liu, and T. Jung, "Verifiable private multi-party computation: Ranging and ranking," *2013 Proceedings IEEE INFOCOM*, pp. 605–609, 2013.

[28] W. A. Department, "Federated ai technology enabler," Website, 2020, https://github.com/FederatedAI/FATE.

[29] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[30] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. ping Huang, A. Dehghantanha, and G. Srivastava, "A survey on security and privacy of federated learning," *Future Gener. Comput. Syst.*, vol. 115, pp. 619–640, 2021.

[31] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Transactions on Information Forensics and Security*, vol. 13, pp. 1333–1345, 2018.

[32] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," *Allerton*, pp. 909–910, 2015.

[33] 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 *CCS*, 2017, pp. 1175–1191.

[34] 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," *ArXiv*, vol. abs/1711.10677, 2017.

[35] B. A. Pearlmutter, "Fast exact multiplication by the hessian," in *Neural computation*, 1994, pp. 147–160.

[36] N. Agarwal, B. Bullins, and E. Hazan, "Second-order stochastic optimization in linear time," *stat*, vol. 1050, p. 15, 2016.

[37] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Advances in Neural Information Processing Systems*, 2019, pp. 14 747–14 756.

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

[39] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *NeurIPS*, 2019.

[40] W. Du, Y. S. Han, and S. Chen, "Privacy-preserving multivariate statistical analysis: Linear regression and classification," in *SDM*, 2004.

[41] G. Si-yang, "Privacy preserving association rule mining in vertically partitioned data," *Journal of Computer Applications*, 2006.

[42] Y. LeCun, "The mnist database," http://yann.lecun.com/exdb/mnist/.

[43] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[44] Kaggle, https://www.kaggle.com/datasets.

[45] P. Zhao and T. Zhang, "Stochastic optimization with importance sampling for regularized loss minimization," in *ICML*, 2015.

[46] A. Vahdat, "Toward robustness against label noise in training deep discriminative neural networks," in *NIPS*, 2017.